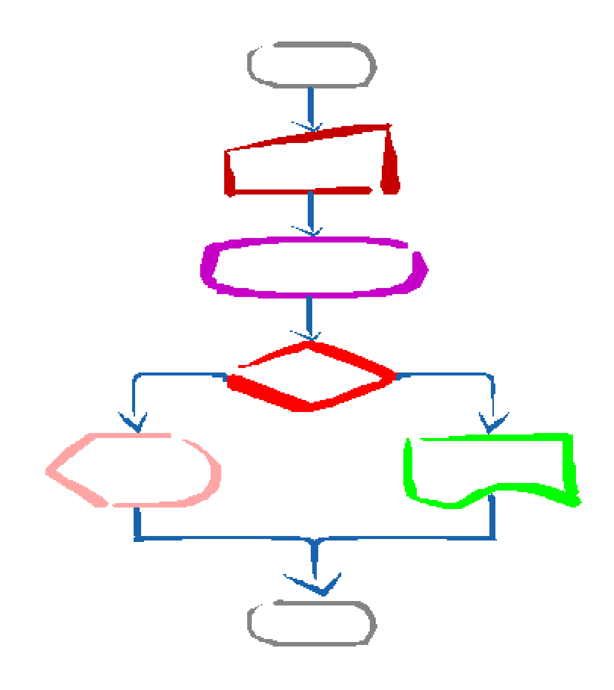
EDUCACIÓN BÁSICA

ALGORITMOS Y PROGRAMACIÓN

GUÍA PARA DOCENTES



JUAN CARLOS LÓPEZ GARCÍA

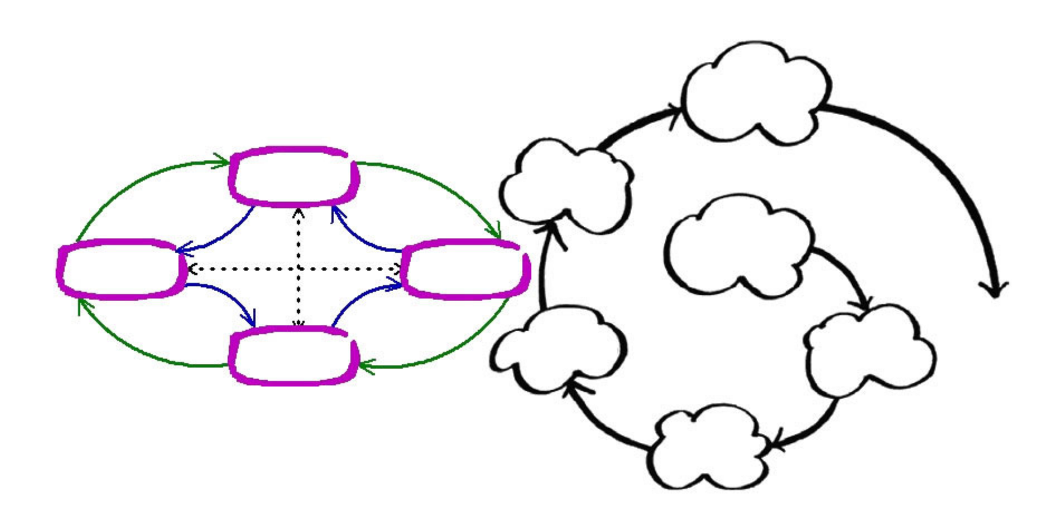
Segunda Edición



Fundación Gabriel Piedrahita Uribe
www.eduteka.org

ALGORITMOS Y PROGRAMACIÓN (GUÍA PARA DOCENTES)
SEGUNDA EDICIÓN, 2007, 2009.

Juan Carlos López García
http://www.eduteka.org



A este documento lo acompaña un Cuaderno de Trabajo para estudiantes que se puede descargar gratuitamente de: http://www.eduteka.org/GuiaAlgoritmos.php

El autor agradece el envío de cualquier comentario sobre esta Guía a los correos: editor@eduteka.org; jualop@gmail.com



La segunda edición de esta Guía se elaboró gracias al apoyo de Motorola Foundation, Motorola de Colombia Ltda. y la gestión de la ONG Give to Colombia.

PROGRAMACIÓN DE COMPUTADORES EN EDUCACIÓN ESCOLAR

UNIDAD 1: DESARROLLO DE HABILIDADES DE PENSAMIENTO

DESARROLLO DE HABILIDADES DE PENSAMIENTO DE ORDEN SUPERIOR

Existe actualmente un consenso general dentro de la comunidad educativa mundial sobre la necesidad de superar el tipo de enseñanza basada en la transmisión de contenidos para apuntarle en su lugar al desarrollo de capacidades. Investigaciones y estudios recientes proponen diversos conjuntos de habilidades que la educación debe fomentar para que los estudiantes puedan tener éxito en el mundo digital y globalizado en el que van a vivir. Este planteamiento exige, sin dilaciones, implementar estrategias que contribuyan efectivamente en el desarrollo de esas habilidades planteadas como fundamentales para la educación en el Siglo XXI (21st Century Skills, 2004).

En la mayoría de conjuntos de habilidades propuestos figuran las habilidades de pensamiento de orden superior entre las que se incluye la destreza para solucionar problemas; por esta razón, se requiere seleccionar estrategias efectivas para ayudar a que los estudiantes las desarrollen. Para atender esta necesidad, la programación de computadores constituye una buena alternativa, siempre y cuando se la enfoque al logro de esta destreza y no a la formación de programadores. Es importante insistir en esta orientación debido a que las metodologías utilizadas en Educación Básica para llevar a cabo cursos de Algoritmos y Programación, son heredadas de la educación superior y muchos de los docentes que las utilizan se dedican principalmente a enseñar los vericuetos de lenguajes de programación profesionales tales como Java, C++, Visual Basic, etc. Hablar hoy de aprender a diseñar y construir aplicaciones (programas) complejas, implica una labor titánica que en la mayoría de los casos está fuera del alcance de la Educación Básica ya que demanda necesariamente enfoques de programación como el orientado a objetos al que apuntan la mayoría de tendencias en Ingeniería de Sistemas.

Por esta razón, en la Educación Básica es altamente recomendable utilizar ambientes de programación basados en Logo, fáciles de utilizar y que permitan realizar procedimientos que contengan estructuras básicas (secuencial, decisión y repetición), pero siempre conducentes al desarrollo de habilidades del Siglo XXI. Solo en los últimos grados de básica secundaria o en la Media Técnica sería aconsejable introducir a los estudiantes a la programación orientada a objetos mediante entornos de programación visuales y amigables como Alice, KPL o Processing.

Desde el punto de vista educativo, la programación de computadores posibilita no solo activar una amplia variedad de estilos de aprendizaje (Stager, 2003) sino desarrollar el pensamiento algorítmico. Adicionalmente, compromete a los estudiantes en la consideración de varios aspectos importantes para la solución de problemas: decidir sobre la naturaleza del problema, seleccionar una representación que ayude a resolverlo y, monitorear sus propios pensamientos (metacognición) y estrategias de solución. Este último, es un aspecto que deben desarrollar desde edades tempranas. No debemos olvidar que solucionar problemas con ayuda del computador puede convertirse en una excelente herramienta para adquirir la costumbre de enfrentar problemas predefinidos de manera rigurosa y sistemática, aun, cuando no se utilice un computador para solucionarlo.

Esto en cuanto a la solución de problemas, pero hay otra habilidad de pensamiento que también se puede ayudar a desarrollar con un cursos de Algoritmos y Programación: La Creatividad.

En los últimos años, la creatividad forma parte de las prioridades de los sistemas educativos en varios países, junto a otras habilidades de pensamiento de orden superior. Al punto que los Estándares Nacionales Norteamericanos de TIC para Estudiantes (NETS-S) formulados en 1998, estaban encabezados por “Operaciones y conceptos básicos de las TIC” y la Creatividad no figuraba. Sin embargo, en la nueva versión de estos Estándares, liberada en 2008, la creatividad encabeza los seis grupos de estándares. Otro ejemplo muy diciente es la creación en Inglaterra del Consorcio para la Creatividad que busca promover en la educación el desarrollo de habilidades de pensamiento que conduzcan la formación de personas orientadas a la creatividad y a la innovación.

Una de las razones para que la creatividad se hubiese convertido en tema prioritario es que tiene un alto impacto en la generación de riqueza por parte de las empresas de la Sociedad de la Creatividad. En esta empresas, los reconocimientos profesionales se dan gracias al talento, la creatividad y la inteligencia. La creatividad reemplazó las materias primas como fuente fundamental de crecimiento económico. Para tener éxito en esta nueva Sociedad, las regiones deben desarrollar, atraer y retener a personas talentosas y creativas que generen innovaciones (Banaji & Burn, 2006). Cada vez es mayor el número de empresas que fundamentan su modelo de negocio en la creatividad y la innovación; para ellas, son indispensables personas que además de tener los conocimientos requeridos para desempeñarse en los diferentes cargos, tengan habilidad para pensar y actuar creativamente.

Ejemplo tangible de esto es el que la Comisión Europea, consciente de la importancia que tienen la creatividad y

la innovación para el desarrollo social y económico de los países del viejo continente, decidiera proclamar el 2009 como el "Año de la Creatividad y la Innovación" (http://create2009.europa.eu/).

El reto enorme que recae hoy sobre los sistemas educativos consiste en lograr que se generen las estrategias adecuadas para que los estudiantes se desarrollen como pensadores creativos. Así como para la sociedad griega en tiempos de Alejandro el Grande era prioridad que las personas desarrollaran su cuerpo como preparación para los quehaceres del campo de batalla, para la sociedad actual es prioritario que las personas desarrollen sus habilidades de pensamiento de orden superior para que pueden desempeñarse con éxito en ella. Pero, dado que el desarrollo de estas habilidades se debe iniciar desde edad temprana, la educación debe asumir su cuota de responsabilidad en esta importante tarea.

Desde este punto de vista, la presente "Guía de Algoritmos y Programación", dirigida a docentes de Educación Básica, se concentra en el desarrollo de la creatividad y de habilidades para solucionar problemas predefinidos. Para facilitar a los docentes su utilización en el aula, los ejemplos que se proponen corresponden a temas de Matemáticas y Ciencias Naturales para grados cuarto y quinto de Básica Primaria.

Programación y Matemáticas

Son varios los temas de las matemáticas cuya comprensión se puede mejorar mediante la integración de esta asignatura con un curso de algoritmos y programación:

- *Concepto de variable*. Una variable es una ubicación de memoria en el computador o en la calculadora que tiene un nombre (identificador) y en la que se pueden almacenar diferentes valores.
- *Concepto de función*. La mayoría de calculadoras científicas vienen de fabrica con cientos de funciones y los estudiantes pueden crear procedimientos que se comportan como funciones (aceptan parámetros, realizan cálculos y reportan un resultado).
- *Manejo de ecuaciones y graficación.*
- *Modelado matemático*. Algunas de las ideas clave de los modelos matemáticos están presentes en los manipulables virtuales (simulaciones y micromundos). Estos manipulables se pueden emplear tanto en procesos de entrenamiento (drill and practice) como de educación matemática. Sin embargo, la tendencia es a utilizarlos en ambientes en los que los estudiantes se convierten en diseñadores y no en simples consumidores.
- *Evaluación*. En la mayoría de las situaciones extraescolares, las personas que necesitan utilizar matemáticas regularmente tienden a usar calculadoras, computadores y otros dispositivos especializados (GPS, medición con láser, etc) como ayuda en la solución de problemas. Esto sugiere que una evaluación autentica en matemáticas debe realizarse con libro y cuaderno abiertos, permitir el uso de calculadora y computador; en cuyo caso el computador puede aportar un ambiente de aprendizaje y evaluación enriquecidos.
- Adicionalmente, hay otros campos más avanzados de las matemáticas que también se pueden impactar con un curso de algoritmos y programación: Inteligencia artificial, robótica, aprendizaje asistido por computador (CAL), aprendizaje asistido por computador altamente interactivo e inteligente (HIICAL), etc.

Es muy importante tener presente que resolver problemas matemáticos mediante procedimientos tiene dos ciclos: uno en el cual se resuelve el problema matemático en sí (con papel y lápiz) y otro en el que esa solución se automatiza en el computador. Crear un procedimiento para calcular el área de cualquier rectángulo a partir de las dimensiones de sus lados, requiere que el estudiante primero resuelva el problema matemático (entender el problema, trazar un plan, ejecutar el plan y revisar) y luego elabore el procedimiento que pida los datos de entrada, realice los cálculos y muestre el resultado (analizar el problema, diseñar un algoritmo, traducir el algoritmo a un lenguaje de programación y depurar el programa).

Programación y Ciencias Naturales

En Ciencias Naturales también hay temas en los cuales realizar actividades de programación de computadores puede ayudar a mejorar su comprensión por parte de los estudiantes.

Mediante el trabajo con entornos de programación como Scratch o MicroMundos, los estudiantes aprenden a seleccionar, crear y manejar múltiples formas de medios (texto, imágenes y grabaciones de audio). La comunicación efectiva requiere hoy en día, para ser creativa y persuasiva, la escogencia y manipulación de los mismos tipos de medios que estos entornos de programación ponen al alcance de los estudiantes. Se espera que a medida que ellos ganan experiencia creando con medios, se vuelvan más perceptivos y críticos en el análisis de los que tienen a su alcance en el mundo que los rodea (Rusk, Resnick & Maloney, 2007).

Por ejemplo, realizar proyectos cuyo producto final sea la comunicación de resultados obtenidos en procesos de indagación y/o experimentación en clase de Ciencias:

- Explicación de las partes de una célula y su importancia como unidad básica de los seres vivos.
- Exposición de los diversos sistemas de órganos del ser humano con la respectiva explicación de su función.
- Clasificación de los seres vivos en diversos grupos taxonómicos (plantas, animales, microorganismos,

etc).
- Descripción y comparación de diversos tipos de neuronas.
- Explicación de las funciones de los diversos componentes de un circuito eléctrico.
- Descripción de los principales elementos del sistema solar que incluya las relaciones de tamaño, movimiento y posición.

Adicionalmente, la elaboración de simulaciones es un veta muy rica para formular proyectos en Ciencias Naturales. Estas sin duda contribuyen a la comprensión de fenómenos naturales ya que en este tipo de actividades los estudiantes actúan como diseñadores. No se debe pasar por alto que los estudiantes aprendan más construyendo materiales de instrucción que estudiándolos (Jonassen, Carr & Yue, 1998).

Para poder construir las simulaciones los estudiantes deben coordinar periodicidad y reglas de interacción entre varios objetos móviles programables (tortugas y objetos). Además, la posibilidad de programar interacciones con el usuario de la simulación ofrece oportunidades valiosas para comprometer al estudiante "diseñador" en la reflexión sobre detección de actividad, retroalimentación, usabilidad y otros elementos presentes en los sistemas de computo que se utilizan diariamente en empresas y hogares.

Los siguientes son algunos ejemplos de proyectos cuyo producto final consiste en una simulación sobre diversos temas que son fundamentales en Ciencias Naturales:

- Imitación del comportamiento de seres vivos en ecosistemas, teniendo en cuenta necesidades y cantidades disponibles de energía y nutrientes (cadena alimentaria).
- Representación del fenómeno migratorio de varias especies de animales como respuesta a cambios en el ambiente y en los ecosistemas en los que viven.
- Ilustración interactiva de la adaptación de los seres vivos a variaciones en el entorno en que viven.
- Demostración del funcionamiento de circuitos eléctricos en serie y en paralelo.
- Imitación del efecto de la transferencia de energía térmica en los cambios de estado de algunas sustancias.
- Representación del ciclo de vida de una planta teniendo en cuenta factores ambientales (semilla – siembra – desarrollo planta – florecimiento – polinización – semilla).
- Diseño de experimentos en los cuales se deba modificar una variable para dar respuesta a preguntas.
- Imitación de fenómenos de camuflaje con un entorno y relacionarlos con ciertas necesidades en algunos seres vivos.
- Representación del movimiento y desplazamiento de objetos con diferentes velocidades.
- Ilustración interactiva del sistema solar.

En el caso de las Ciencias Naturales es fundamental tener en cuenta que los proyectos que se realicen utilizando entornos de programación como Scratch o MicroMundos pueden requerirse conocimiento básico del tema por parte de los estudiantes. En este caso, los proyectos se convierten en profundización para la comprensión de esos temas.

De lo contrario, se debe incluir en cada proyecto un componente previo de investigación en el cual los estudiantes puedan alcanzar los conocimientos básicos conceptuales requeridos para llevar a cabo el proyecto. La mayoría de estos proyectos de este tipo se enfocan en la comunicación de hallazgos de indagaciones y/o experimentaciones.

Una tercera opción es trabajar con el entorno de programación como actividad exploratoria a un tema dado en Ciencias Naturales; sin embargo, este enfoque requiere una planeación cuidadosa para que se logre el objetivo de aprendizaje y la actividad no se quede únicamente en lo lúdico.

SOLUCIÓN DE PROBLEMAS

Una de las acepciones que trae el Diccionario de Real Academia de la Lengua Española (RAE) respecto a la palabra Problema es "Planteamiento de una situación cuya respuesta desconocida debe obtenerse a través de métodos científicos". Con miras a lograr esa respuesta, un problema se puede definir como una situación en la cual se trata de alcanzar una meta y para lograrlo se deben hallar y utilizar unos medios y unas estrategias. La mayoría de problemas tienen algunos elementos en común: un **estado inicial**; una **meta**, lo que se pretende lograr; un conjunto de **recursos**, lo que está permitido hacer y/o utilizar; y un **dominio**, el estado actual de conocimientos, habilidades y energía de quien va a resolverlo (Moursund, 1999).

Casi todos los problemas requieren, que quien los resuelve, los divida en **submetas** que, cuando son dominadas (por lo regular en orden), llevan a alcanzar el objetivo. La solución de problemas también requiere que se realicen **operaciones** durante el estado inicial y las submetas, actividades (conductuales, cognoscitivas) que alteran la naturaleza de tales estados (Schunk, 1997).

Cada disciplina dispone de estrategias específicas para resolver problemas de su ámbito; por ejemplo, resolver problemas matemáticos implica utilizar estrategias propias de las matemáticas. Sin embargo, algunos psicólogos opinan que es posible utilizar con éxito estrategias generales, útiles para resolver problemas en muchas áreas. A través del tiempo, la humanidad ha utilizado diversas estrategias generales para resolver problemas. Schunk (1997), Woolfolk (1999) y otros, destacan los siguientes métodos o estrategias de tipo general:

- ***Ensayo y error*** : Consiste en actuar hasta que algo funcione. Puede tomar mucho tiempo y no es seguro que se llegue a una solución. Es una estrategia apropiada cuando las soluciones posibles son pocas y se pueden probar todas, empezando por la que ofrece mayor probabilidad de resolver el problema. Por ejemplo, una bombilla que no prende: revisar la bombilla, verificar la corriente eléctrica, verificar el interruptor.
- ***Iluminación*** : Implica la súbita conciencia de una solución que sea viable. Es muy utilizado el modelo de cuatro pasos formulado por Wallas (1921): preparación, incubación, iluminación y verificación. Estos cuatro momentos también se conocen como proceso creativo. Algunas investigaciones han determinado que cuando en el periodo de incubación se incluye una interrupción en el trabajo sobre un problema se logran mejores resultados desde el punto de vista de la creatividad. La incubación ayuda a "olvidar" falsas pistas, mientras que no hacer interrupciones o descansos puede hacer que la persona que trata de encontrar una solución creativa se estanque en estrategias inapropiadas.
- ***Heurística*** : Se basa en la utilización de reglas empíricas para llegar a una solución. El método heurístico conocido como "IDEAL", formulado por Bransford y Stein (1984), incluye cinco pasos: Identificar el problema; definir y presentar el problema; explorar las estrategias viables; avanzar en las estrategias; y lograr la solución y volver para evaluar los efectos de las actividades (Bransford & Stein, 1984). El matemático Polya (1957) también formuló un método heurístico para resolver problemas que se aproxima mucho al ciclo utilizado para programar computadores. A lo largo de esta Guía se utilizará este método propuesto por Polya.
- ***Algoritmos*** : Consiste en aplicar adecuadamente una serie de pasos detallados que aseguran una solución correcta. Por lo general, cada algoritmo es específico de un dominio del conocimiento. La programación de computadores se apoya en este método, tal como veremos en la Unidad 2.
- ***Modelo de procesamiento de información*** : El modelo propuesto por Newell y Simon (1972) se basa en plantear varios momentos para un problema (estado inicial, estado final y vías de solución). Las posibles soluciones avanzan por subtemas y requieren que se realicen operaciones en cada uno de ellos.
- ***Análisis de medios y fines*** : Se funda en la comparación del estado inicial con la meta que se pretende alcanzar para identificar las diferencias. Luego se establecen submetas y se aplican las operaciones necesarias para alcanzar cada submeta hasta que se alcance la meta global. Con este método se puede proceder en retrospectiva (desde la meta hacia el estado inicial) o en prospectiva (desde el estado inicial hacia la meta).
- ***Razonamiento analógico*** : Se apoya en el establecimiento de una analogía entre una situación que resulte familiar y la situación problema. Requiere conocimientos suficientes de ambas situaciones.
- ***Lluvia de ideas*** : Consiste en formular soluciones viables a un problema. El modelo propuesto por Mayer (1992) plantea: definir el problema; generar muchas soluciones (sin evaluarlas); decidir los criterios para estimar las soluciones generadas; y emplear esos criterios para seleccionar la mejor solución. Requiere que los estudiantes no emitan juicios con respecto a las posibles soluciones hasta que terminen de formularlas.
- ***Sistemas de producción*** : Se basa en la aplicación de una red de secuencias de condición y acción (Anderson, 1990).
- ***Pensamiento lateral*** : Se apoya en el pensamiento creativo, formulado por Edwar de Bono (1970), el cual difiere completamente del pensamiento lineal (lógico). El pensamiento lateral requiere que se exploren y consideren la mayor cantidad posible de alternativas para solucionar un problema. Su importancia para la educación radica en permitir que el estudiante: explore (escuche y acepte puntos de

vista diferentes, busque alternativas); avive (promueva el uso de la fantasía y del humor); libere (use la discontinuidad y escape de ideas preestablecidas); y contrarreste la rigidez (vea las cosas desde diferentes ángulos y evite dogmatismos). Este es un método adecuado cuando el problema que se desea resolver no requiere información adicional, sino un reordenamiento de la información disponible; cuando hay ausencia del problema y es necesario apercibirse de que hay un problema; o cuando se debe reconocer la posibilidad de perfeccionamiento y redefinir esa posibilidad como un problema (De Bono, 1970).

Como se puede apreciar, hay muchas estrategias para solucionar problemas; sin embargo, esta Guía se enfoca principalmente en dos de estas estrategias: Heurística y Algorítmica.

Según Polya (1957), cuando se resuelven problemas, intervienen cuatro operaciones mentales:

1. Entender el problema
2. Trazar un plan
3. Ejecutar el plan (resolver)
4. Revisar

Numerosos autores de textos escolares de matemáticas hacen referencia a estas cuatro etapas planteadas por Polya. Sin embargo, es importante notar que estas son flexibles y no una simple lista de pasos como a menudo se plantea en muchos de esos textos (Wilson, Fernández & Hadaway, 1993). Cuando estas etapas se siguen como un modelo lineal, resulta contraproducente para cualquier actividad encaminada a resolver problemas.

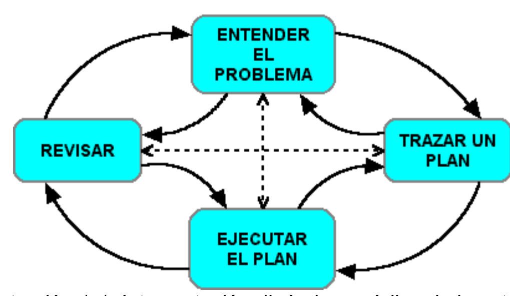


Ilustración 1-1: Interpretación dinámica y cíclica de las etapas planteadas por Polya para resolver problemas.

Es necesario hacer énfasis en la naturaleza dinámica y cíclica de la solución de problemas. En el intento de trazar un plan, los estudiantes pueden concluir que necesitan entender mejor el problema y deben regresar a la etapa anterior; o cuando han trazado un plan y tratan de ejecutarlo, no encuentran cómo hacerlo; entonces, la actividad siguiente puede ser intentar con un nuevo plan o regresar y desarrollar una nueva comprensión del problema (Wilson, Fernández & Hadaway, 1993; Guzdial, 2000).

TIP

La mayoría de los textos escolares de matemáticas abordan la Solución de Problemas bajo el enfoque planteado por Polya. Por ejemplo, en "Recreo Matemático 5" (Díaz, 1993) y en "Dominios 5" (Melo, 2001) se pueden identificar las siguientes sugerencias propuestas a los estudiantes para llegar a la solución de un problema matemático:

1. ***COMPRENDER EL PROBLEMA.***
   - Leer el problema varias veces
   - Establecer los datos del problema
   - Aclarar lo que se va a resolver (¿Cuál es la pregunta?)
   - Precisar el resultado que se desea lograr
   - Determinar la incógnita del problema
   - Organizar la información
   - Agrupar los datos en categorías
   - Trazar una figura o diagrama.
2. ***HACER EL PLAN.***
   - Escoger y decidir las operaciones a efectuar.
   - Eliminar los datos inútiles.
   - Descomponer el problema en otros más pequeños.
3. ***EJECUTAR EL PLAN (Resolver).***
   - Ejecutar en detalle cada operación.
   - Simplificar antes de calcular.
   - Realizar un dibujo o diagrama
4. ***ANALIZAR LA SOLUCIÓN (Revisar).***
   - Dar una respuesta completa
   - Hallar el mismo resultado de otra manera.
   - Verificar por apreciación que la respuesta es adecuada.

EJEMPLO

En un juego, el ganador obtiene una ficha roja; el segundo, una ficha azul; y el tercero, una amarilla. Al final de varias rondas, el puntaje se calcula de la siguiente manera: Al cubo de la cantidad de fichas rojas se adiciona el doble de fichas azules y se descuenta el cuadrado de las fichas amarillas. Si Andrés llegó 3 veces en primer lugar, 4 veces de último y 6 veces de intermedio, ¿Qué puntaje obtuvo? (Adaptado de Melo (2001), página 30).

R/.

COMPRENDE

- Leer detenidamente el problema
- ¿Cuántos colores de fichas se reparten?
- ¿Cuántas fichas rojas, azules y amarillas obtuvo Andrés?
- ¿Qué pregunta el problema?

PLANEA

- Para hallar el puntaje que obtiene Andrés por sus llegadas de primero, calcular el cubo de la cantidad de fichas rojas.
- Para hallar el puntaje por sus llegadas en segundo lugar, calcular el doble de la cantidad de fichas azules.
- Para hallar el puntaje que pierde por sus llegadas en último lugar, calcular el cuadrado de la cantidad de fichas amarillas.
- Para hallar el puntaje total, calcular la suma de los puntajes por las fichas rojas y azules, restarle los puntos de las fichas amarillas.

RESUELVE

- Por tres fichas rojas: $3^3 = 27$ puntos
- Por seis fichas azules: $6 \times 2 = 12$ puntos
- Por cuatro fichas amarillas: $4^2 = 16$ puntos
- Para obtener el puntaje final de Andrés, sumar los puntos obtenidos con las fichas rojas y azules (27 + 12 = 39 puntos) y de este resultado restar los puntos representados por las fichas amarillas (39 – 16 = 23 puntos).

REVISA

- El puntaje que obtuvo Andrés es 23 puntos.
- Verificar las operaciones y comparar los cálculos con la solución estimada.

El anterior es un problema típico en clase de matemáticas. Es muy importante que los estudiantes reflexionen sobre las actividades que realizan para solucionarlo (metacognición) y las agrupen de acuerdo a las etapas que contenga la estrategia de solución empleada.

ACTIVIDAD

En la academia de las ciencias sociales hay dos grupos de materias: Geografía, con 124 alumnos; Historia, con 220; y Educación Ambiental, con 185. Si hay 25 alumnos que estudian Geografía y Educación Ambiental, 37 que estudian Educación Ambiental e Historia, y ninguno toma las tres materias, ¿cuántos alumnos tiene la academia? (Adaptado de Melo, 2001, página 46).

El estudiante debe tener en cuenta (y anotar) las actividades que realiza para resolver este problema y agruparlas en cada una de las cuatro etapas propuestas por Polya (comprende, planea, resuelve y revisa). Para resolver este problema, los estudiantes deben tener conocimientos sobre conjuntos (representación, clasificación e intersección). Es buena idea que construyan una tabla para organizar la información y un diagrama de Venn para representar los datos.

Establecer un modelo para solucionar problemas es un paso fundamental pero no suficiente. Según Clements & Meredith (1992) y Zemelman, Daniels & Hyde (1998) y otros, los docentes deben adoptar una serie de buenas prácticas con el fin de ayudar a los estudiantes a desarrollar habilidades para resolver problemas:

- Plantear verbalmente problemas con variedad de estructuras y de formas de solución.
- Presentar diversas estrategias de solución de problemas.
- Asignar problemas que tengan aplicación en la vida diaria.
- Ofrecer experiencias que estimulen la curiosidad de los estudiantes y construyan confianza en la investigación, la solución de problemas y la comunicación.
- Permitir a los estudiantes tomar la iniciativa en el planteamiento de preguntas e investigaciones que les interesen.
- Hacer preguntas que involucren pensamiento de orden superior.
- Verificar que los estudiantes son conscientes de las estrategias que deben utilizar y de los procesos que deben aprender.
- Plantear problemas que proporcionen contextos en los que se aprendan conceptos y habilidades.
- Proveer ejemplos de cómo los conceptos y habilidades utilizados podrían aplicarse en otros contextos.
- Promover, de manera creciente, la abstracción y la generalización mediante la reflexión y la experimentación.
- Fomentar la utilización de representaciones visuales que favorezcan la comprensión de conceptos (diagramas de flujo, mapas conceptuales, diagramas de Venn, etc).
- Dar retroalimentación personalizada en consideración al esfuerzo hecho por los estudiantes para solucionar problemas.
- Verificar que una cantidad importante de la instrucción ocurra en grupos pequeños o en situaciones de uno a uno.
- Ventilar los errores y malentendidos más comunes.
- Promover la interacción tanto estudiante-docente, como estudiante-estudiante. Los niños son los mejores maestros de otros niños en cosas tan importantes para ellos como el aprendizaje de diversos juegos (Savater, 1996).
- Ofrecer actividades que den oportunidad a los estudiantes de discutir, hacer conjeturas, sacar conclusiones, defender ideas y escribir conceptualizaciones.
- Proporcionar oportunidades para realizar trabajo reflexivo y colaborativo entre estudiantes.

Solución de problemas y programación

Desde el punto de vista educativo, la solución de problemas mediante la programación de computadores posibilita la activación de una amplia variedad de estilos de aprendizaje. Los estudiantes pueden encontrar diversas maneras de abordar problemas y plantear soluciones, al tiempo que desarrollan habilidades para: visualizar caminos de razonamiento divergentes, anticipar errores, y evaluar rápidamente diferentes escenarios mentales (Stager, 2003).

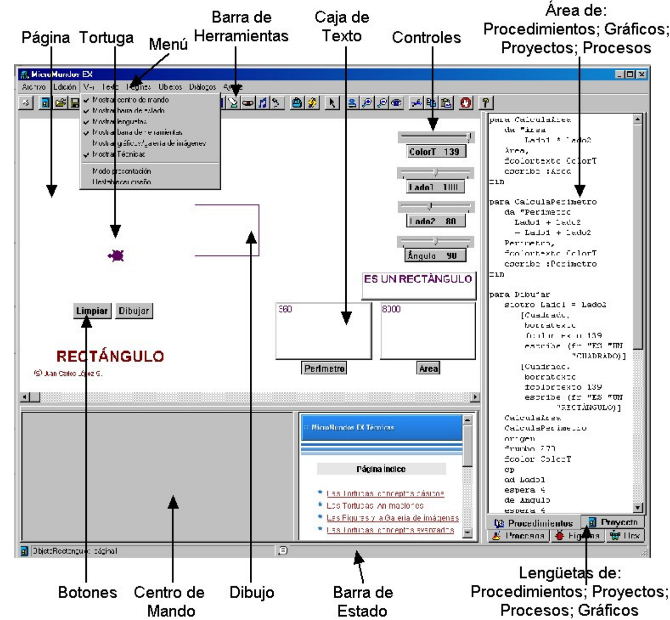


Ilustración 1-2(a): Área de trabajo de MicroMundos EX (interfaz del programa)

Quienes han utilizado Logo con estudiantes de básica primaria (especialmente con grados 3º a 5º - 8 a 11

años) habrán podido observar la facilidad con que ellos se familiarizan con la interfaz del programa y la utilizan para darle instrucciones a la tortuga. Por ejemplo, utilizan el “centro de mando” (área de comandos) para introducir manualmente, una a una, las instrucciones para construir un rectángulo. Esta forma de utilizar Logo promueve la **exploración** y permite al estudiante ver inmediatamente cuál es el efecto que produce cada instrucción ejecutada.

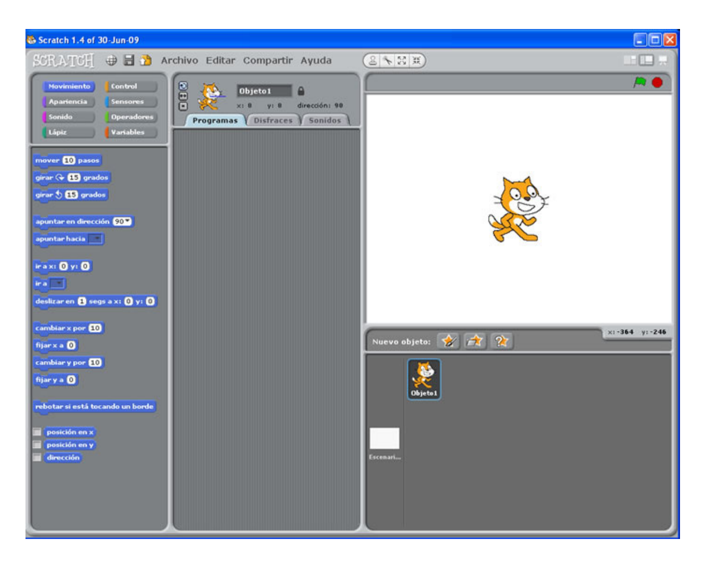


Ilustración 1-2(b): Área de trabajo de Scratch Versión 1.4 (interfaz del programa)

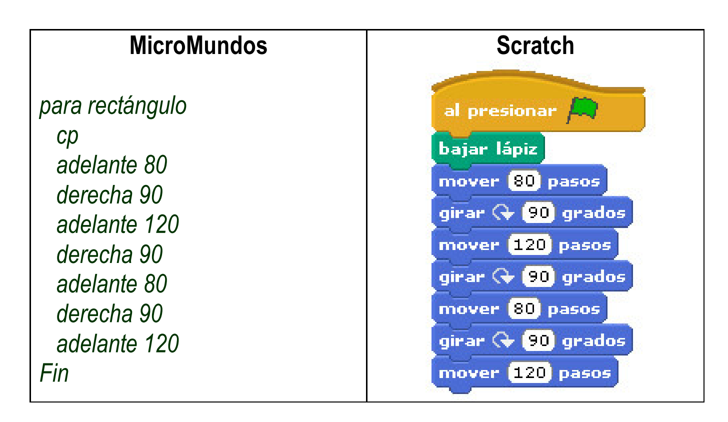
EJEMPLO
Pedir a los estudiantes que escriban en el “Centro de Mando” las instrucciones para dibujar un rectángulo con las siguientes medidas: Lado1= 80; Lado2=120.

MicroMundos	Scratch
cp *adelante 80* *derecha 90* *adelante 120* *derecha 90* *adelante 80* *derecha 90* *adelante 120*	*El Centro de Mando de MicroMundos no tiene equivalente en Scratch.*

A medida que el estudiante introduce cada una de estas instrucciones se dibuja cada uno de los lados que conforman el rectángulo.
NOTA: Ver en el Anexo 1 un resumen de las primitivas (comandos e instrucciones) de MicroMundos y de Scratch utilizadas en esta guía.

Sin embargo, en esta guía se utilizará el “área de procedimientos” de MicroMundos para programar el computador. Los procedimientos son módulos con instrucciones que se inician con el comando “para” y que el computador ejecuta automáticamente, una tras otra, hasta encontrar el comando “fin”. Emplear Logo de esta manera exige que el estudiante piense en todos los comandos que conforman un procedimiento antes de escribirlo, ejecutarlo y comprobar si produce el resultado esperado. Así, Logo promueve lo que Piaget (1964) denominó “la conquista de la difícil conducta de la **reflexión**” que se inicia a partir de los siete u ocho años cuando niños y niñas dejan de actuar por impulso y empiezan a pensar antes de proceder. Además, demanda de los estudiantes planificar, formular hipótesis y anticipar qué sucederá.

EJEMPLO
Pedir a los estudiantes que escriban un procedimiento para dibujar un rectángulo con unas medidas determinadas (Lado1= 80; Lado2=120), implica que ellos deben pensar en algo muy parecido a lo siguiente (y escribirlo):



MicroMundos	Scratch
para rectángulo *cp* *adelante 80* *derecha 90* *adelante 120* *derecha 90* *adelante 80* *derecha 90* *adelante 120* *Fin*	al presionar bajar lápiz mover 80 pasos girar 90 grados mover 120 pasos girar 90 grados mover 80 pasos girar 90 grados mover 120 pasos

Cuando se invoca este procedimiento escribiendo “*rectángulo*” en el “Centro de Mando” de MicroMundos o haciendo clic en la bandera verde de Scratch, el computador ejecuta automáticamente y en orden consecutiva, las instrucciones que se encuentran entre “*para rectángulo*” [*to rectangulo*] y “*fin*” [*end*] (MicroMundos) o debajo de la instrucción [al presionar bandera verde]. Antes de escribir el anterior procedimiento, los estudiantes deben analizar la figura geométrica que desean construir, describirla y reflexionar acerca de cómo se unen sus partes (dos pares de lados paralelos de igual longitud y cuatro ángulos iguales de 90 grados). Deben explicar el todo mediante la composición de las partes, y esta composición supone, por tanto, la existencia de autenticas operaciones de segmentación o partición y de operaciones inversas de reunión o adición, así como desplazamientos por separación o concentración (Piaget, 1964).

Pedir a los estudiantes que escriban un procedimiento más general para dibujar cualquier rectángulo, significa que ellos deben tratar las dimensiones de la figura como variables (Lado1= ?; Lado2= ?) y no como constantes (Lado1= 80; Lado2= 120). Además, deben construir una definición de rectángulo que el computador entienda; de esta manera, empiezan a construir conocimiento intuitivo acerca de la definición de esta figura geométrica, conocimiento que luego pueden formalizar en una definición abstracta de la misma (Clements & Meredith, 1992).

Adicionalmente, la programación de computadores compromete a los estudiantes en varios aspectos importantes de la solución de problemas: decidir sobre la naturaleza del problema, seleccionar una representación que les ayude a resolverlo, y monitorear sus propios pensamientos (metacognición) y estrategias

de solución. Este último, es un aspecto que ellos deben desarrollar desde edades tempranas y solucionar problemas con ayuda del computador puede convertirse en una excelente herramienta para adquirir la costumbre de tratar cualquier problema de manera rigurosa y sistemática, aun, cuando no se vaya a utilizar un computador para solucionarlo.

De hecho, para muchos educadores, el uso apropiado de la tecnología en la educación tiene un significado similar a la solución de problemas matemáticos. La programación de computadores para llevar a cabo tareas matemáticas retadoras puede mejorar la comprensión del estudiante "programador" sobre las matemáticas relacionadas con una solución. Esto implica abrirle un espacio a la programación en el estudio de las matemáticas, pero enfocándose en los problemas matemáticos y en el uso del computador como una herramienta para solucionar problemas de esta área (Wilson, Fernández & Hadaway, 1993).

Numerosos autores de libros sobre programación, plantean cuatro fases para elaborar un procedimiento que realice una tarea específica. Estas fases concuerdan con las operaciones mentales descritas por Polya para resolver problemas:

1. Analizar el problema (*Entender el problema*)
2. Diseñar un algoritmo (*Trazar un plan*)
3. Traducir el algoritmo a un lenguaje de programación (*Ejecutar el plan*)
4. Depurar el programa (Revisar)

Como se puede apreciar, hay una similitud entre las metodologías propuestas para solucionar problemas matemáticos (Clements & Meredith, 1992; Díaz, 1993; Melo, 2001; NAP, 2004) y las cuatro fases para solucionar problemas específicos de áreas diversas, mediante la programación de computadores.

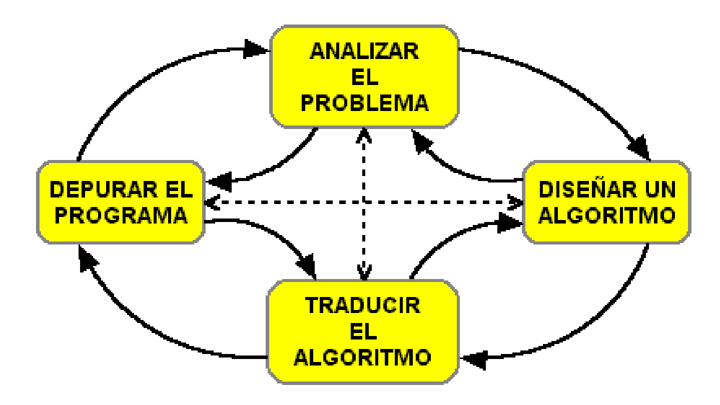


Ilustración 1-3: fases para elaborar un programa de computador.

Analizar el problema (entenderlo)

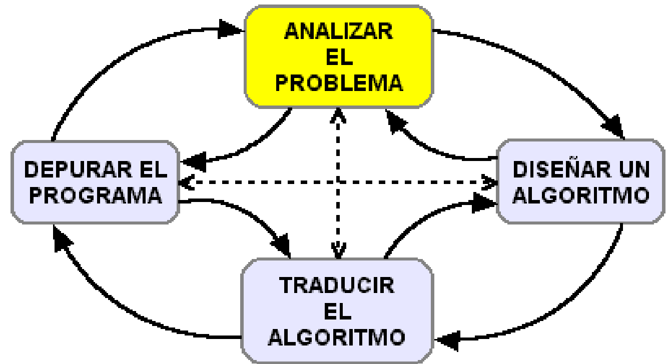


Ilustración 1-4: Primera fase del ciclo de programación.

Los programas de computador tienen como finalidad resolver problemas específicos y el primer paso consiste en definir con precisión el problema hasta lograr la mejor comprensión posible. Una forma de realizar esta actividad se basa en **formular claramente el problema**, especificar los **resultados** que se desean obtener, identificar la **información disponible** (datos), determinar las **restricciones** y definir los **procesos** necesarios para convertir los datos disponibles (materia prima) en la información requerida (resultados).

Estas etapas coinciden parcialmente con los elementos generales que, según Schunk (1997), están presentes en todos los problemas:

1. Especificar claramente los resultados que se desean obtener (*meta y submetas*)
2. Identificar la información disponible (*estado inicial*)
3. Definir los procesos que llevan desde los datos disponibles hasta el resultado deseado (*operaciones*)

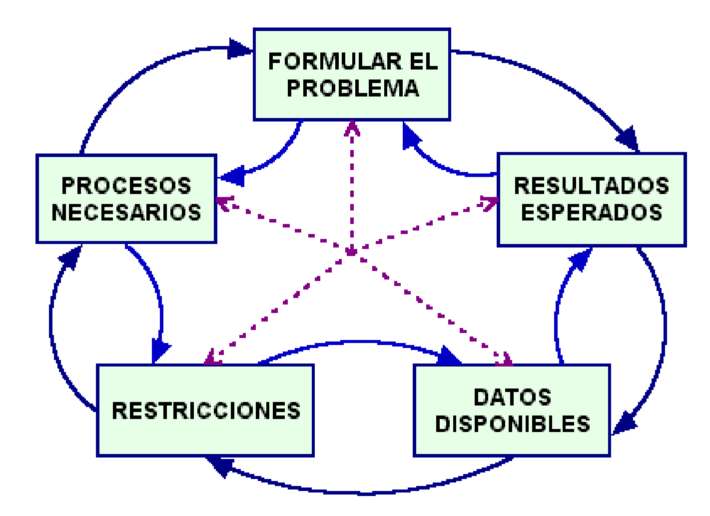


Ilustración 1-5: Etapas a desarrollar en la fase de análisis de un problema (entenderlo)

Para establecer un modelo que los estudiantes puedan utilizar en la fase de análisis del problema, debemos agregar dos temas a los elementos expuestos por Schunk (1997): formular el problema y determinar las restricciones.

Ahora veamos con mayor detalle cada una de las etapas del análisis de un problema.

Formular el problema

La solución de un problema debe iniciar por determinar y comprender exactamente en qué consiste ese problema. La mayoría de los problemas que se resuelven en el aula de clase llegan a manos de los estudiantes perfectamente formulados. Esta etapa es una buena oportunidad para plantear situaciones en forma verbal o escrita que vinculen la enseñanza de las matemáticas con el entorno en el que vive el estudiante y que tengan una variedad de estructuras y de formas de solución (Zemelman, Daniels & Hayde, 1998).

Esta metodología obliga al estudiante a formular el problema a partir de la situación real planteada. De esta manera se contrarresta la costumbre tan común en el aula de que los problemas sean formulados por el profesor o tomados de los libros de texto (Brown & Walter, 1990).

EJEMPLO
OPCIÓN 1:
Juan Felipe es jefe de bodega en una fabrica de pañales desechables y sabe que la producción diaria es de 744 pañales y que en cada caja donde se empacan para la venta caben 12 pañales. ¿Cuántas cajas debe conseguir Juan Felipe para empacar los pañales fabricados en una semana?
OPCIÓN 2:
Juan Felipe es jefe de bodega en una fabrica de pañales desechables y una de las tares del día consiste en llamar al proveedor de los empaques y ordenarle la cantidad suficiente de cajas para empacar los pañales fabricados en la semana próxima. El jefe de producción le informó ayer a Juan Felipe que la producción diaria será de 744 pañales y en cada caja cabe una docena de ellos. ¿Qué debe hacer Felipe?

La Opción 1 plantea directamente el problema que el estudiante debe resolver. Mientras que la Opción 2 plantea una situación y la pregunta es ¿Qué debe hacer Felipe?. La Opción 2 demanda al estudiante leer muy bien el texto para comprender la situación y así poder formular el problema de Juan Felipe. Es algo similar a preguntar al estudiante "cuánto es 7 menos 3" versus preguntar "sí Rosa tiene 7 naranjas y Julio tiene 3, cuántas naranjas de más tiene Rosa".

La comprensión lingüística del problema (entender el significado de cada enunciado) es muy importante. El estudiante debe realizar una lectura previa del problema con el fin de obtener una visión general de lo que se le pide y una segunda lectura para poder responder preguntas como:

- ¿Puedo definir mejor el problema?
- ¿Qué palabras del problema me son desconocidas?
- ¿Cuáles son las palabras clave del problema?
- ¿He resuelto antes algún problema similar?
- ¿Qué información es importante?
- ¿Qué información puedo omitir?

Además, es conveniente que los estudiantes se habitúen a analizar los problemas desde diferentes puntos de vista y a categorizar la información dispersa que reciben como materia prima (Schunk, 1997).

En programación es frecuente que quien programa deba formular el problema a partir de los resultados esperados. Es muy importante que el estudiante sea consciente de que cuando las especificaciones de un programa se comunican mediante lenguaje natural, estas pueden ser ambiguas, incompletas e incongruentes. En esta etapa se debe hacer una representación precisa del problema (Rumbaugh, 1996); especificar lo más exactamente posible lo que hay que hacer (no cómo hay que hacerlo).

EJEMPLO
Doña Ruby necesita decidir cómo comprar un televisor que cuesta 850.000 de contado o 960.000 a crédito. Ella tiene 600.000 pesos en efectivo.
R/.
Como el efectivo que tiene doña Ruby no le alcanza para comprar el televisor de contado, ella tiene dos opciones: comprarlo totalmente a crédito o pagar una parte de contado (cuota inicial) y el resto a crédito.
Para poder resolver el problema se debe conocer el número de cuotas si desea pagarlo totalmente a crédito o conocer el número de cuotas y el valor total del televisor si se da una cuota inicial de 600.000 pesos.

Precisar los resultados esperados
(*meta y submetas*)

Para establecer los resultados que se esperan (meta) es necesario identificar la información relevante, ignorar los detalles sin importancia, entender los elementos del problema y activar el esquema correcto que permita comprenderlo en su totalidad (Woolfolk, 1999).

Determinar con claridad cuál es el resultado final (producto) que debe devolver el programa es algo que ayuda a establecer la meta. Es necesario analizar qué resultados se solicitan y qué formato deben tener esos resultados (impresos, en pantalla, diagramación, orden, etc). El estudiante debe preguntarse:

- ¿Qué información me solicitan?
- ¿Qué formato debe tener esta información?

Identificar datos disponibles
(*estado inicial*)

Otro aspecto muy importante en la etapa de análisis del problema consiste en determinar cuál es la información disponible. El estudiante debe preguntarse:

- ¿Qué información es importante?
- ¿Qué información no es relevante?
- ¿Cuáles son los datos de entrada? (conocidos)
- ¿Cuál es la incógnita?
- ¿Qué información me falta para resolver el problema? (datos desconocidos)

- ¿Puedo agrupar los datos en categorías?

Otro aspecto importante del estado inicial hace referencia al nivel de conocimiento que el estudiante posee en el ámbito del problema que está tratando de resolver. Es conveniente que el estudiante se pregunte a sí mismo:

- ¿Qué conocimientos tengo en el área o áreas del problema?
- ¿Son suficientes esos conocimientos?
- ¿Dónde puedo obtener el conocimiento que necesito para resolver el problema?
- ¿Mis compañeros de estudio me pueden ayudar a clarificar mis dudas?
- ¿Qué expertos en el tema puedo consultar?

En el ámbito de las matemáticas, se conoce como conocimiento condicional a aquel que activan los estudiantes cuando aplican procedimientos matemáticos concretos de manera intencional y consciente a ciertas situaciones. "El conocimiento condicional proporciona al alumno un sistema de valoración sobre la extensión y las limitaciones de su saber (qué sabe sobre el tema, su capacidad de memoria, etc), a la vez que examina la naturaleza de la demanda del profesor y su objetivo último, y evalúa variables externas como pueden ser el tiempo que tiene o con quién realiza la tarea" (Orubia & Rochera & Barberà, 2001).

EJEMPLO
Esteban está ahorrando para comprar una patineta que vale 55.000 pesos. Su papá le ha dado una mesada de 5.000 pesos durante 7 semanas. Por lavar el auto de su tío tres veces recibió 8.000 pesos. Su hermano ganó 10.000 pesos por hacer los mandados de su mamá y 4.000 por sacar a pasear el perro. ¿Esteban tiene ahorrado el dinero suficiente para comprar la patineta o aún le falta? (Adaptado de Casasbuenas & Cifuentes (1998b), página 23).
R/.
Formular el problema: Ya se encuentra claramente planteado.
Resultados esperados: Si o no tiene Esteban ahorrado el dinero suficiente para comprar una patineta que vale 55.000 pesos.
Datos disponibles: Los ingresos de Esteban: 5.000 pesos por 7 semanas + 8.000 pesos. Los 10.000 y 4.000 pesos qué ganó el hermano de Esteban son irrelevantes para la solución de este problema y se pueden omitir.

Determinar las restricciones

Resulta fundamental que los estudiantes determinen aquello que está permitido o prohibido hacer y/o utilizar para llegar a una solución. En este punto se deben exponer las necesidades y restricciones (no una propuesta de solución). El estudiante debe preguntarse:

- ¿Qué condiciones me plantea el problema?
- ¿Qué está prohibido hacer y/o utilizar?
- ¿Qué está permitido hacer y/o utilizar?
- ¿Cuáles datos puedo considerar fijos (constantes) para simplificar el problema?
- ¿Cuáles datos son variables?
- ¿Cuáles datos debo calcular?

Establecer procesos (***operaciones***)

Consiste en determinar los procesos que permiten llegar a los resultados esperados a partir de los datos disponibles. El estudiante debe preguntarse:

- ¿Qué procesos necesito?
- ¿Qué fórmulas debo emplear?
- ¿Cómo afectan las condiciones a los procesos?
- ¿Qué debo hacer?
- ¿Cuál es el orden de lo que debo hacer?

En la medida de lo posible, es aconsejable dividir el problema original en otros más pequeños y fáciles de solucionar (submetas), hasta que los pasos para alcanzarlas se puedan determinar con bastante precisión (módulos). Esto es lo que en programación se denomina diseño descendente o top-down (Joyanes, 2001).

El diseño descendente se utiliza en la programación estructurada de computadores debido a que facilita:

- La comprensión del problema
- Las modificaciones en los módulos
- La verificación de la solución

Al realizar divisiones sucesivas del problema en otros más pequeños y manejables (módulos), hay que tener cuidado para no perder de vista la comprensión de este como un todo. El estudiante, luego de dividir el problema original en submetas (módulos), debe integrar cada parte de tal forma que le permita comprender el problema como un todo (Woolfolk, 1999).

Igualmente hay que tener cuidado cuando se utiliza este enfoque para resolver problemas complejos o extensos, en cuyo caso resulta más aconsejable utilizar una metodología orientada a objetos. Especialmente, cuando profesores universitarios manifiestan su preocupación por el aprendizaje de malas prácticas de programación en el colegio. Hay casos en los cuales algunos estudiantes no han podido cambiar su forma de pensar "estructurada" por otra orientada a objetos, la cual hace parte de los programas universitarios modernos en la carrera de Ingeniería de Sistemas. Es aconsejable que los ejemplos y actividades planteados a los estudiantes contengan solo un problema cuya solución sea muy corta (no necesariamente sencillo de resolver). De esta forma ellos podrán enfocarse en aplicar completamente la metodología propuesta para analizar problemas (formular el problema, especificar los resultados, identificar la información disponible, determinar las restricciones y definir los procesos) sin perderse en el laberinto de un problema demasiado complejo.

Las operaciones para llegar a los resultados esperados se implementan en Logo mediante procedimientos. Por ejemplo, si se desea producir un software para trabajar con figuras geométricas de diferentes tipos, el triángulo rectángulo será uno de los objetos a tener en cuenta y este a su vez, debe prestar los siguientes servicios (Jiménez, 2002):

1. Un procedimiento para leer los datos de entrada.
2. Un procedimiento para calcular el área.
3. Un procedimiento para calcular la hipotenusa.
4. Un procedimiento para calcular el perímetro.
5. Un procedimiento para mostrar los resultados.

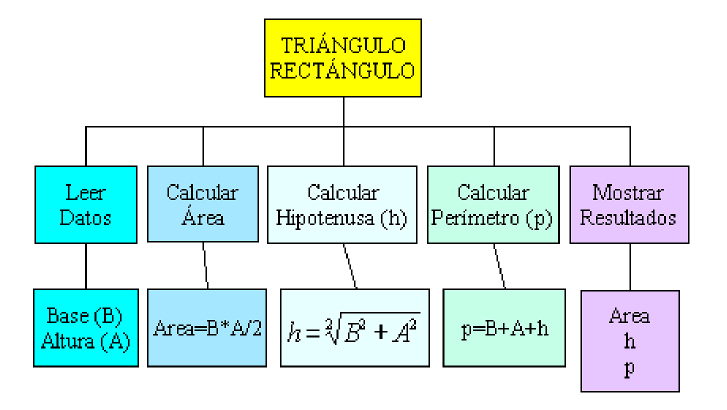


Ilustración 1-6: Descripción de los servicios que debe estar en capacidad de prestar el objeto "triángulo rectángulo".

EJEMPLO
De acuerdo con la metodología descrita, analizar el problema de hallar el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.
R/
Formular el problema: Ya se encuentra claramente planteado.
Resultados esperados: El área de un triángulo rectángulo.
Datos disponibles: Base, Altura, Hipotenusa, tipo de triángulo. La incógnita es el área y todos los valores son constantes. El valor de la hipotenusa se puede omitir. El estudiante debe preguntarse si sus conocimientos actuales de matemáticas le permiten resolver este problema; de no ser así, debe plantear una estrategia para obtener los conocimientos requeridos.
Determinar las restricciones: Utilizar las medidas dadas.
Procesos necesarios: Guardar en dos variables los valores de Base y Altura; Guardar en una constante el divisor 2; aplicar la fórmula área=base*altura/2; comunicar el resultado (área).

ACTIVIDAD
La mayoría de las metodologías propuestas para la solución de problemas matemáticos se aproxima al ciclo de programación de computadores. Se puede iniciar planteando a los estudiantes problemas matemáticos como los siguientes, encontrados en Casasbuenas & Cifuentes (1998b):
1. *Luisa quiere invertir sus ahorros en la compra de discos compactos de moda. Si tiene $68.000, ¿Cuántos discos comprará?*

Analizar el problema:
- ¿Qué tienes en cuenta cuando vas a comprar un disco?
- ¿Tienes información suficiente para resolver el problema de Luisa?
- ¿Qué dato averiguarías para saber cuántos discos puede comprar Luisa?

Plantear ahora este problema utilizando la metodología de "Formular el problema", "Resultados esperados", "Datos disponibles", "Determinar las restricciones" y "Procesos necesarios".

TIP
Cinco pasos que deben tener en cuenta los estudiantes para resolver problemas matemáticos (Rodríguez, 1995):
1. Leer con mucho cuidado el problema hasta entenderlo.
2. Buscar la(s) pregunta(s).
3. Decidir lo que debes hacer.
4. Realizar las operaciones.
5. Comprobar que la respuesta hallada es correcta.

Pida a los estudiantes que contesten las siguientes preguntas en el proceso de solución de problemas matemáticos:
- ¿Cuántas preguntas tiene el problema? ¿Cuáles?
- ¿Qué debes hacer primero? ¿Para qué?
- ¿Qué debes hacer luego? ¿Para qué?
- ¿Cuál debe ser la respuesta (estimada) del problema?

ACTIVIDAD
Basándose en la metodología expuesta en esta unidad, dividir a los estudiantes en grupos y distribuir entre ellos la tarea de análisis detallado ("Formular el problema", "Resultados esperados", "Datos disponibles", "Determinar las restricciones" y "Procesos necesarios") de los siguientes problemas (uno por grupo):
1. Hallar el área de un cuadrado cuyo lado mide 5 cm.
2. Hallar uno de los lados de un rectángulo cuya área es de 15 cm^2 y uno de sus lados mide 3 cm.
3. Hallar el área y el perímetro de un círculo cuyo radio mide 2 cm.
4. Hallar el área de un pentágono regular de 6 cm de lado y con 4 cm de apotema.

Dato Curioso
Deep Blue de IBM fue el primer computador que superó a un campeón mundial de ajedrez cuando le ganó una partida a Gary Kasparov en febrero de 1996. La victoria de Deep Blue formaba parte de una serie de seis partidas, que Kasparov terminó ganando 4-2. En 1997, una versión nueva y mejorada de Deep Blue contraatacó en una segunda serie. Esta vez, el computador, capaz de planear una vertiginosa cantidad de 200 millones de posiciones por segundo, ganó la serie a Kasparov por 3.5 a 2.5 puntos. (Libro Gunness de los Records 2002)

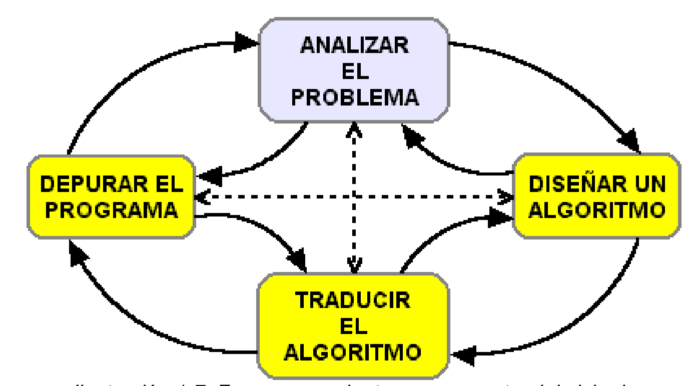


Ilustración 1-7: Fases segunda, tercera y cuarta, del ciclo de programación.

Diseñar el algoritmo (trazar un plan)
Este tema se tratará en profundidad en las unidades 2 y 3 de esta guía. Por el momento, podemos resumir que únicamente hasta cuando se ha realizado un análisis a

fondo del problema (utilizando alguna metodología), se puede proceder a elaborar el algoritmo (diagrama de flujo). Este consiste en la representación gráfica, mediante símbolos geométricos, de la secuencia lógica de las instrucciones (plan) que posteriormente serán traducidas a un lenguaje de programación, como Logo, para ejecutarlas y probarlas en un computador.

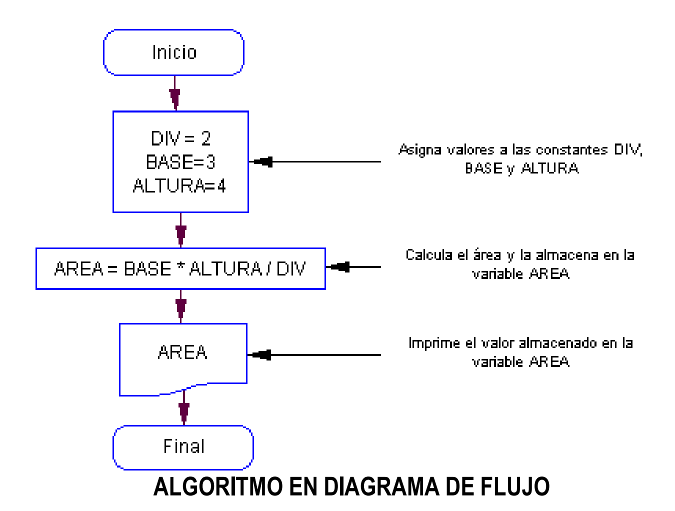
EJEMPLO
Diseñar un algoritmo (seudocódigo y diagrama de flujo) para hallar el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.
R/

ANÁLISIS DEL PROBLEMA

Formular el problema: Ya se encuentra claramente planteado.
Resultados esperados: El área de un triángulo rectángulo.
Datos disponibles: Base, Altura, Hipotenusa, tipo de triángulo. La incógnita es el área y todos los valores son constantes. El valor de la hipotenusa se puede omitir. El estudiante debe preguntarse si sus conocimientos actuales de matemáticas le permiten resolver este problema; de no ser así, debe plantear una estrategia para obtener los conocimientos requeridos.
Determinar las restricciones: Utilizar las medidas dadas.
Procesos necesarios: Guardar en dos variables (BASE y ALTURA) los valores de Base y Altura; Guardar en una constante (DIV) el divisor 2; aplicar la fórmula BASE*ALTURA/DIV y guardar el resultado en la variable AREA; comunicar el resultado (AREA).

ALGORITMO EN SEUDOCÓDIGO

Paso 1: Inicio
Paso 2: Asignar el número 2 a la constante "div"
Paso 3: Asignar el número 3 a la constante "base"
Paso 4: Asignar el número 4 a la constante "altura"
Paso 5: Guardar en la variable "área" el resultado de base*altura/div
Paso 6: Imprimir el valor de la variable "área"
Paso 7: Final



ALGORITMO EN DIAGRAMA DE FLUJO

Ilustración 1-8: Diagrama de Flujo para hallar el área de un triángulo rectángulo.

Traducir el algoritmo (ejecutar el plan)

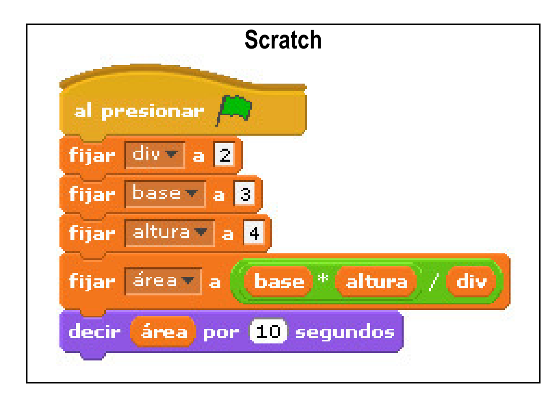
Este tema se tratará en profundidad en las Unidades 3 y 4 de esta guía. Una vez que el algoritmo está diseñado y representado gráficamente se pasa a la etapa de traducción a un lenguaje de programación determinado (en nuestro caso será Logo). Cada lenguaje posee sus propias reglas gramaticales, por lo tanto es fundamental que los estudiantes conozcan de antemano la sintaxis de los comandos que deben utilizar para resolver el problema. A mayor dominio del lenguaje de programación, mayor posibilidad de llegar rápidamente a una solución satisfactoria. A esta fase de traducción se le conoce comúnmente como codificación.

EJEMPLO
A partir del ejemplo anterior, escribir un procedimiento en Logo que se llame triángulo para hallar el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.
R/

MicroMundos

```
para triángulo
  local "div
  local "base
  local "altura
  local "área
  da "div 2
  da "base 3
  da "altura 4
  da "área :base * :altura / :div
  muestra :área
fin
```

Scratch



Al escribir en el centro de mando de MicroMundos la palabra *triángulo* se debe obtener como resultado 6. En el caso de Scratch, hacer clic en la bandera verde y se debe obtener el mismo resultado.

Depurar el programa (revisar)

Este tema se tratará en profundidad en la Unidad 4 de esta guía. Después de traducir el algoritmo en un lenguaje de programación como Logo, el programa resultante debe ser probado y validados los resultados. A este proceso se le conoce como depuración. Depurar programas contribuye a mejorar la capacidad en los estudiantes para resolver problemas; la depuración basada en la retroalimentación es una habilidad útil para toda la vida (Stager, 2003).

Quienes han escrito alguna vez un programa de computador, saben de la dificultad que representa elaborar programas perfectos en el primer intento,

dificultad que aumenta a medida que el problema a resolver es más complejo. La depuración, afinamiento y documentación de un programa hacen parte fundamental del ciclo de programación y desde el punto de vista educativo estimula en los estudiantes la curiosidad, la perspectiva, la comunicación y promueve valores como responsabilidad, fortaleza, laboriosidad, paciencia y perseverancia. La programación facilita un diálogo interior en el cual la retroalimentación constante y el éxito gradual empujan a los alumnos a ir más allá de sus expectativas (Stager, 2003).

Otras dos actividades relacionadas con esta etapa son la afinación y la documentación. La primera consiste en realizar retoques para lograr una mejor apariencia del programa (en pantalla o en los resultados impresos) o para ofrecer funcionalidades más allá de los resultados esperados (especificados en la fase de análisis del problema). La segunda tiene un carácter eminentemente comunicativo, con la documentación de un programa se pone a prueba la capacidad del estudiante para informar a otras personas cómo funciona su programa y lo que significa cada elemento utilizado.

EJEMPLO

Complementar la solución del problema de hallar el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.

R/

La base y la altura son suficientes para calcular el área de un triángulo rectángulo (resultado esperado), pero adicionalmente se puede calcular el perímetro (afinación), aplicando la fórmula:

perímetro=Base+Altura+Hipotenusa

Incluso, en caso que el enunciado del problema no hubiese indicado el valor de la Hipotenusa, si se poseen los suficientes conocimientos de geometría, se puede calcular el valor de esta a partir de la Base, la Altura y la condición de ser un triángulo rectángulo:

$$Hipotenusa = \sqrt[2]{Base^2 + Altura^2}$$

Dato Curioso

Spacewar es el primer videojuego del mundo. Se empezó a utilizar en 1961 en el Massachusetts Institute of Technology (MIT) en un computador PDP-1. Se trataba de un juego de combate espacial en el que dos naves alrededor de una estrella central debían derribarse entre ellas. Programado como diversión por estudiantes del MIT , este juego fue el precursor de todos los videojuegos modernos. El computador PDP-1 se puso a la venta en 1960 y costaba 120.000 dólares (el equivalente a 930.000 dólares actuales) y en total se vendieron 50 unidades. El PDP-1 es el antepasado del computador personal actual y se concibió para su uso en instituciones científicas. Disponía de una memoria de 4Kb y los operadores empleaban un teclado y cinta de papel perforado para la introducción de datos. (Libro Guinness de los Records, 2002).

Si se quiere llegar a un planteamiento, para Educación Básica, que contribuya efectivamente a desarrollar la creatividad programando computadores, es conveniente como primera medida, llegar a un acuerdo sobre qué es la creatividad, pues varios autores la definen de manera diferente.

De acuerdo con el Diccionario de la Real Academia Española (RAE), creatividad es la facultad de crear o la capacidad de creación. Por su parte, la enciclopedia Microsoft Encarta define la Creatividad como la capacidad de inventar algo nuevo, de relacionar algo conocido de forma innovadora o de apartarse de los esquemas de pensamiento y conducta habituales. Según Wikipedia, la creatividad es un proceso mental y social que implica generar nuevas ideas o conceptos, o nuevas asociaciones entre ideas y conceptos conocidos, lo que habitualmente produce soluciones originales. Las definiciones anteriores se refieren al acto de inventar cualquier cosa nueva (Ingenio), a la capacidad de encontrar soluciones originales y a la voluntad de modificar o transformar el mundo.

Ana Craft (2001) anota que las definiciones de creatividad más aceptadas en los últimos 50 años son aquellas que unen creatividad e imaginación. Este enfoque sugiere que cada persona tiene potencial creativo ya que este es un aspecto fundamental de la naturaleza humana. Ella se refiere a la “creatividad con c minúscula” como la habilidad para hacer frente, de manera efectiva, a los retos y cambios que nos plantea la vida en el siglo XXI. Esta es la creatividad que sirve para afrontar tareas cotidianas (elaborar una nueva receta o un arreglo floral, escribir una carta o poema, enseñar un nuevo truco a alguien, etc). También entra en juego cuando se deben superar obstáculos tales como desempleo y pobreza o aprovechar oportunidades. Esta “creatividad” se contrapone a la “Creatividad con C mayúscula” propuesta por el psiquiatra Gene Cohen (citado por Banaji & Burn, 2006), que caracteriza los logros extraordinarios de personas poco corrientes como artistas renombrados, científicos e inventores.

Stenberg (1997), autor reconocido en este campo, argumenta que la creatividad no es solo una capacidad, sino un proceso en el que intervienen tres tipos de inteligencia: creativa (ir más allá de lo dado y engendrar ideas nuevas e interesantes), analítica (analizar y evaluar ideas, resolver problemas y tomar decisiones) y práctica (traducir teorías abstractas en realizaciones efectivas). Estas dos últimas inteligencias aportan la posibilidad de diferenciar entre ideas innovadoras buenas y malas y, además, relacionarlas con la vida cotidiana (López, 2000). Por su parte, Gardner (1993) define a la persona creativa como alguien que “regularmente resuelve problemas, genera productos o define nuevos cuestionamientos en un dominio, de manera que en principio se considera nueva pero que al final llega a ser aceptada por un grupo cultural particular”.

En los Estándares Nacionales Estadounidenses de TIC para Estudiantes (NETS'S), reformulados por ISTE, el primer grupo corresponde a Creatividad e Innovación. Para ISTE, los estudiantes al finalizar sus Educación Media deben demostrar pensamiento creativo, construir conocimiento y desarrollar productos y procesos innovadores utilizando las TIC.

Según ISTE (2007), los estudiantes deben estar en capacidad de aplicar su conocimiento previo para generar nuevas ideas, productos o procesos; crear trabajos originales como medios de expresión personal o grupal; usar modelos y simulaciones para explorar sistemas y temas complejos; e identificar tendencias y prever posibilidades.

Según el Comité Consultivo Nacional para la Educación Creativa y Cultural de Inglaterra (NACCCE, por su sigla en Inglés), la creatividad se define como la actividad imaginativa que tiene como objetivo producir resultados tanto originales como generadores de valor (Robinson, 1999).

Para el Consorcio de Habilidades de Aprendizaje para el Siglo XXI, las habilidades de aprendizaje e innovación se están reconociendo como aquellas que separan a los estudiantes que están preparados para los ambientes de vida y de trabajo del Siglo XXI, cada vez más complejos, de los que no lo están. Hacer énfasis en creatividad, pensamiento crítico, comunicación y colaboración es esencial en la preparación de los estudiantes para el futuro. Entre las competencias de creatividad e innovación que propone el Consorcio están: demostrar originalidad e inventiva en el trabajo; desarrollar, implementar y comunicar nuevas ideas a otros; tener apertura y responder a perspectivas nuevas y diversas; y actuar con ideas creativas para realizar una contribución tangible y útil en el campo en el que ocurre la innovación.

Por su parte, el Consorcio para la Creatividad propone que ésta se refiere a mucho más que “hacer arte”. La creatividad tiene que ver con el desarrollo de la capacidad para: cuestionar, hacer conexiones, innovar, resolver problemas y reflexionar críticamente; todas éstas son habilidades altamente valoradas en el mundo laboral actual; y agregan, “el aprendizaje creativo empodera a los jóvenes a imaginar un mundo diferente y les da confianza y motivación para llevar a cabo lo que imaginan” (Creative Partnerships, 2006).

Son muchas las definiciones que intentan explicar el concepto de creatividad, aquí solo se exponen algunas de ellas a fin de dar una perspectiva amplia a los docentes en este campo. El desarrollo de pensamiento algorítmico que promueve esta guía, mediante el

enfoque de solución de problemas predefinidos, se complementa con el desarrollo de pensamiento creativo. Pues en el mundo actual, en el que lo único permanente es el cambio, además de aprender a resolver tipos específicos de problemas, los estudiantes deben aprender a improvisar creativamente cuando se encuentren con situaciones inesperadas y a explorar alternativas de solución variadas (Resnick, 2007).

Desarrollo de la creatividad

Una de las cuestiones en torno a la creatividad que aún no tiene respuesta definitiva es si esta se puede desarrollar o simplemente se nace con dicha "genialidad" (C mayúscula).

Respecto al genio creativo, el Consorcio para la Creatividad considera que éste es un discurso posromántico apoyado por quienes han visto la creatividad únicamente como una cualidad especial de pocas personas, generalmente artistas, tales como escritores, músicos, pintores, etc (Banaji & Burn, 2006). Los que explican la creatividad desde una perspectiva basada en características de la personalidad afirman que las personas creativas tienen rasgos comunes: buen humor; confianza en sí mismos; flexibilidad y adaptabilidad; alta capacidad de asociación; sensibilidad; curiosidad intelectual; percepción y observación agudas; iniciativa para tomar riesgos; imaginación; expresividad; capacidad crítica; entusiasmo; y, tenacidad (López, 2000). Por el contrario, quienes no son creativos presentan recurrentemente algunas de los siguientes rasgos: tienden a especializarse en ciertos temas; son extremadamente racionales; les falta confianza en si mismos; no tienen motivación; su capacidad para escuchar es reducida; respetan la autoridad en exceso; no son buenos observadores; y, tienen deficiente pensamiento crítico.

Sin embargo, buena parte de los autores que han trabajado en profundidad el tema de la creatividad, entre ellos Resnick, De Bono y Johansson, no solo argumentan que si es posible desarrollarla, sino que aportan propuestas concretas para trabajarla en el aula de clase. Además, plantean que las siguientes habilidades cognitivas, susceptibles de desarrollar, están presentes en las personas consideradas como creativas: se plantean nuevos objetivos; exploran un mayor número de alternativas; evalúan, durante el transcurso del proceso de solución, los objetivos, las alternativas y las tareas; se aseguran de entender a cabalidad los problemas; son observadores; usan la abstracción; usan metáforas y analogías; desglosan la tarea en subtareas y desarrollan productos intermedios; y, usan estrategias metacognitivas (López, 2000).

Según De Bono (1970), es conveniente empezar a enseñar, a partir de los 7 años, técnicas de pensamiento que faciliten el desarrollo de la creatividad. Entre las que se pueden implementar en cursos de diferentes asignaturas tenemos: plantear problemas inesperados, formular alternativas, proponer e implementar diseños, realizar observaciones, hacer abstracción en diversos temas, realizar ejercicios de dibujo y utilizar metáforas y analogías.

Sin embargo, puede sonar ambicioso implementar toda una metodología para desarrollar la creatividad en un curso de Algoritmos y Programación. Por esto y para efectos de la presente Guía, se seleccionaron dos técnicas de pensamiento que hacen una contribución al desarrollo de la creatividad: planteamiento de problemas inesperados y formulación de alternativas.

El planteamiento de problemas inesperados busca complementar el enfoque de solución de problemas predefinidos que para resolverlos pueden hacer uso de metodologías como la propuesta por Polya. Es precisamente esta metodología la que se ha utilizado para resolver problemas matemáticos; sin embargo, algunos docentes han manifestado preocupación ya que si bien, la metodología ayuda a que los estudiantes estructuren su pensamiento, muchos de ellos se encasillan en ella y les cuesta trabajo encontrar soluciones alternativas.

En este sentido, entornos de programación como Scratch y MicroMundos, comprometen a los estudiantes en la búsqueda de soluciones innovadoras a problemas inesperados; no se trata solamente de aprender a solucionar problemas de manera predefinida, sino de estar preparado para generar nuevas soluciones a medida que los problemas se presentan (Resnick, 2007).

Por su parte, formular alternativas, se basa en el primer principio básico del Pensamiento Lateral propuesto por De Bono (1970): "cualquier modo de valorar una situación es sólo uno de los muchos modos posibles de valorarla". La búsqueda de alternativas a una situación o problema parece un proceso típico del pensamiento lógico; sin embargo, desde el punto de vista de la creatividad no se busca la mejor alternativa sino la formulación del mayor número posible de alternativas. Por lo tanto, es conveniente fijar de entrada y poner por escrito, el número de alternativas que los estudiantes deben plantear. Desde la lógica, por lo general la búsqueda se interrumpe cuando se halla una alternativa que parece satisfactoria.

Como aprestamiento a la realización de proyectos que busquen deliberadamente desarrollar la creatividad, es deseable que los estudiantes realicen actividades tales como: hacer asociación de ideas sobre temas ya vistos en clase, elaborar listados de atributos de objetos cotidianos, buscar al menos 30 usos para cada uno de los objetos cotidianos propuestos, Jugar con Torres de Hanoi de tres y cuatro discos y, elaborar figuras con el Tangram, entre otras. Estas actividades permiten evidenciar el estilo de pensamiento predominante de cada estudiante. Quienes piensan convergentemente tenderán a abordar los problemas de forma lógica, ordenada y a establecer relaciones comunes; quienes piensan divergentemente, tenderán a hacer juicios

ilógicos, innovadores y poco comunes.

Espiral del pensamiento creativo

Con el fin de promover el desarrollo de la creatividad, esta Guía propone utilizar la Espiral del Pensamiento Creativo propuesta por Mitchel Resnick (2007). En esta, los estudiantes imaginan lo que quieren hacer; crean un proyecto basado en sus ideas; juegan con sus ideas y creaciones; comparten sus ideas y creaciones con otros y reflexionan sobre sus experiencias; lo anterior los lleva a imaginar nuevas ideas y nuevos proyectos. La espiral genera un proceso indefinido de mejoramiento continuo.

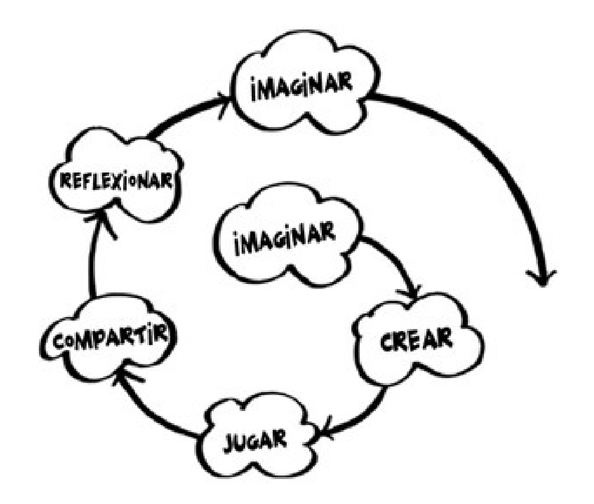


Ilustración 1-9: Espiral del Pensamiento Creativo diseñada por el Dr. Mitchel Resnick

En un comienzo, este proceso lo debe planear y dirigir el docente. Sin embargo, a medida que los estudiantes lo interiorizan, aprenden a recorrerla de manera independiente para desarrollar sus propias ideas, ponerlas a prueba, desafiar límites y fronteras, experimentar con alternativas, recibir retroalimentación de otros y generar nuevas ideas con base en sus experiencias (Resnick, 2007).

Es muy importante que al diseñar las diferentes fases de un proyecto, en el cual los estudiantes utilizarán un ambiente de programación como Scratch o MicroMundos, los docentes tengan en cuenta los elementos de la espiral de la creatividad. Por ejemplo, se deben incluir en el proyecto espacios para compartir el trabajo realizado, para escuchar y valorar la retroalimentación del grupo a cada trabajo individual y para reflexionar sobre las posibles mejoras que se pueden realizar con base en la retroalimentación recibida.

Para que la Espiral del Pensamiento Creativo funcione, los docentes deben promover un ambiente en el que se permita imaginar, transformar, idealizar, desestructurar y reestructurar. Un ambiente donde se pueda comunicar, donde haya tolerancia para las reacciones espontáneas (López, 2000). Los docentes deben propiciar un ambiente de confianza, en el que sea más importante la cantidad de alternativas de solución que generen los estudiantes a un problema planteado, que las respuestas correctas. Por tanto, debe evitarse a toda costa, la emisión de juicios de valor negativos ante cualquier alternativa, por ilógica que parezca.

Adicionalmente, en las diferentes fases de la Espiral, se debe aprovechar cualquier oportunidad para plantear problemas inesperados y para solicitar a los estudiantes que formulen alternativas de solución a cada problema o situación que se presente. En este mismo sentido, la práctica indica que en reiteradas ocasiones los estudiantes plantean a sus docentes situaciones que ellos desean desarrollar en sus proyectos. Dichas situaciones se convierten en problemas inesperados que los docentes deben resolver. Estas situaciones las pueden aprovechar los docentes para plantearlas a toda la clase como problemas inesperados.

EJEMPLO

Proyecto: La cadena alimentaria

Estándares MEN que se cubren (Colombia): Explico la dinámica de un ecosistema teniendo en cuenta las necesidades de energía y nutrientes de los seres vivos (cadena alimentaria).

Descripción: En este proyecto los estudiantes deben representar el comportamiento de varios seres vivos en su respectivo ecosistema, teniendo en cuenta tanto necesidades como cantidades disponibles de energía y nutrientes (cadena alimentaria). Para ello, deben elaborar una simulación, en MicroMundos o en Scratch, de una cadena alimentaria teniendo en cuenta seres productores, hervivoros, carnívoros y omnivoros.

Fases del proyecto:

1. Los estudiantes deben imaginar un ecosistema que contenga por lo menos cuatro seres vivos. Luego deben dibujar o importar los seres vivios que imaginaron. Además, deben dibujar el escenario que representa el ecosistema.
   En este punto, el docente debe estimular la reflexión para que ellos verifiquen que el ecosistema que dibujaron corresponde con los seres vivientes que en la simulación incluyeron en este. Un problema inesperado puede plantearse mediante la pregunta ¿todos los seres vivos que representaste viven en ese ecosistema? Si la respuesta es negativa, deben plantear por escrito al menos tres alternativas de solución (por ejemplo: cambiar el ecosistema, cambiar alguno de los seres vivos, cambiarlo todo, etc).
   Además, cada ser vivo debe tener un tamaño proporcional en relación a los otros seres y elementos del ecosistema.
   En caso de ser necesario, se debe destinar un lapso de tiempo de la clase para investigar ,en Internet o en la Biblioteca Escolar, qué seres vivos habitan en el ecosistema que dibujaron.
2. Esta fase inicia con otro problema inesperado: "La tarea quedó mal planteada y hay que corregirla, de los cuatro seres vivos que se crearon en la fase anterior, debe haber por lo menos un ser vivo de cada tipo: productor, herbívoro, carnívoro y omnivoro".
   Los estudiantes deben investigar qué seres vivos de cada tipo

habitan en el ecosistema que dibujaron. Luego, dibujar o importar los seres vivos correctos para que se cumpla la condición planteada en la tarea rectificada. No es necesario que eliminen los seres vivos que habían creado en la fase 1, siempre y cuando correspondan al ecosistema.

3. A continuación, deben programar el desplazamiento de los seres vivos por todo el espacio disponible y de manera aleatoria (se pueden utilizar las instrucciones "rebotar si está tocando un borde" y "número al azar entre 1 y 15 grados" como parámetro de la instrucción girar).
   Como problema inesperado pedir que se restrinja el movimiento de manera que se aproxime al comportamiento real de cada ser vivo. Por ejemplo, si el fondo tiene tierra y firmamento, entonces un ser vivo que no vuela, no se puede mover en el área de la pantalla que representa el firmamento. Agregar variables para controlar la velocidad a la que se desplaza cada ser vivo. Algunos, como las plantas, tendrán velocidad 0.
4. Los estudiantes comparten el trabajo realizado con el resto de la clase y reciben retroalimentación tanto de sus compañeros, como del docente.
5. Atender la retroalimentación suministrada. Hacer la programación correspondiente para que cuando a un ser vivo lo toque otro al que le sirve de alimento, el primero desaparezca (como si se lo hubiera comido).
6. El problema inesperado ahora es que debe programarse la aparición de varios seres vivos iguales, ubicados en diferentes posiciones de la pantalla (por ejemplo, si uno de los seres vivos de la animación es un conejo, copiar entonces el objeto conejo, al menos tres veces, pues en un ecosistema rara vez se encuentra un solo animal de cada especie). Solicitar a los estudiantes, al menos dos alternativas, para realizar esta tarea. Las apariciones deben hacerse de acuerdo a una tasa de reproducción establecida para cada uno de los seres vivos. Por ejemplo, se reproducen más rápidamente los conejos que los zorros.
   Se puede destinar un tiempo de la clase para investigar la tasa de reproducción de cada uno de los seres vivos que se incluyen en la animación.
7. Compartir con el resto de la clase el trabajo realizado y recibir retroalimentación de los compañeros.
8. Realizar los últimos ajustes al funcionamiento de la simulación y agregar controles para manipular las tasas de reproducción y/o la velocidad de desplazamiento para cada ser vivo.
9. Socializar con el resto de la clase el trabajo finalizado.

En este ejemplo hay que prestar atención a lo siguiente:

En la fase 3, se debe crear una variable por cada ser vivo, que controle la velocidad a la cual este se va a mover (los que no se desplazan, como las plantas, deben inicializarse con valor 0). Por lo regular, esta condición se implementa con el comando "esperar x segundos"; sin embargo, nótese que si el valor de la variable aumenta, en lugar de aumentar la velocidad, lo que hace es disminuirla ya que el tiempo de espera será mayor. Para limitar el desplazamiento de un ser vivo a cierta región de la pantalla, se debe hacer un control permanente (dentro de un por siempre) con el comando "posición y de objeto1" ó "posición x de objeto 1". Si el ser supera ese valor, entonces se lo desplaza en una posición menor en el eje x o y, y se gira 45 grados.

En la fase 6 es recomendable crear cada uno de los seres vivos (ejemplo, el conejo) y hacerle toda la programación para que se comporte de acuerdo a lo esperado. Una vez funcione correctamente la programación, se copia varias veces el ser vivo (ejemplo, el conejo) y se ubica en posiciones diferentes del escenario. Además, cada copia del ser vivo debe aparecer en momentos diferentes para simular la tasa de reproducción.

Por último, bien sea que los estudiantes utilicen el entorno de programación en Ciencias Naturales para comunicar resultados obtenidos en procesos de indagación y/o experimentación o para elaborar simulaciones de diversos fenómenos naturales, los docentes deben promover comportamientos personales y sociales fundamentales para el funcionamiento de la Espiral del Pensamiento Creativo, tales como:

- Escuchar activamente a compañeros y compañeras.
- Reconocer puntos de vista diferentes y compararlos con los propios.
- Reconocer y aceptar el escepticismo de los demás compañeros ante la información que se presenta.
- Cumplir con las funciones asignadas cuando se trabaja en grupo.
- Respetar y cuidar los seres vivos y objetos presentes en el entorno.

.

UNIDAD 2: ALGORITMOS, CONCEPTOS BÁSICOS

¿QUÉ ES UN ALGORITMO?

Ilustración 2-1: Segunda fase del ciclo de programación.

Luego de analizar detalladamente el problema hasta entenderlo completamente, se procede a diseñar un algoritmo (trazar un plan) que lo resuelva por medio de pasos sucesivos y organizados en secuencia lógica. El concepto intuitivo de algoritmo (procedimientos y reglas) se puede encontrar en procesos naturales de los cuales muchas veces no se es conciente. Por ejemplo, el proceso digestivo es un concepto intuitivo de algoritmo con el que se convive a diario sin que haga falta un definición "matemática" del mismo. Tener claro el proceso digestivo, no implica que los alimentos consumidos nutran más. La familiaridad de lo cotidiano impide a las personas ver muchos algoritmos que se suceden a su alrededor. Procesos, rutinas o biorritmos naturales como la gestación, las estaciones, la circulación sanguínea, los ciclos cósmicos, etc, son algoritmos naturales que generalmente pasan desapercibidos.

La rama del saber que mayor utilización ha hecho del enfoque algorítmico es las matemáticas. Durante miles de años el ser humano se ha esforzado por abstraer la estructura de la solución de problemas con el fin de determinar claramente cuál es el camino seguro, preciso y rápido que lleva a esas soluciones. Son abundantes los ejemplos: máximo común divisor, teorema de Pitágoras, áreas de figuras geométricas, división, suma de números fraccionarios, etc. Todos estos algoritmos matemáticos independizan los datos iniciales del problema de la estructura de su solución, lo que permite su aplicación con diferentes conjuntos de datos iniciales (variables).

EJEMPLO

Consideremos el algoritmo de Euclides para hallar el Máximo Común Divisor (MCD) de dos números enteros positivos dados. Obsérvese que no se especifica cuáles son los dos números, pero si se establece claramente una restricción: deben ser enteros y positivos.

ALGORITMO EN SEUDOCÓDIGO

Paso 1: Inicio.
Paso 2: Leer los dos números ("a" y "b"). Avanzar al paso 3.
Paso 3: Comparar "a" y "b" para determinar cuál es mayor. Avanzar al paso 4.
Paso 4: Si "a" y "b" son iguales, entonces ambos son el resultado esperado y termina el algoritmo. En caso contrario, avanzar al paso 5.
Paso 5: Si "a" es menor que "b", se deben intercambiar sus valores. Avanzar al paso 6; si "a" no es menor que "b", avanzar al paso 6.
Paso 6: realizar la operación "a" menos "b", asignar el valor de "b" a "a" y asignar el valor de la resta a "b". Ir al paso 3.

Investigaciones realizadas en Educación Básica (en ambientes constructivistas) recomiendan incluir la solución de problemas en el currículo de matemáticas de forma que provea oportunidades a los estudiantes para crear sus propios algoritmos y generalizarlos a un conjunto específico de aplicaciones (Wilson, Fernández & Hadaway, 1993). Los estudiantes deben reflexionar sobre sus habilidades de planificación y sobre cómo pueden utilizar esas habilidades en diferentes contextos. Por otra parte, en un estudio sobre Logo (Clements & Meredith, 1992), se concluye que cuando los maestros enfatizaron en la elaboración de un plan para desarrollar un procedimiento matemático (este incluía el uso de estrategias como dividir conceptos grandes en otros más pequeños) encontraron que los estudiantes empezaron a utilizar con mayor frecuencia estrategias de planificación y de dibujo para resolver problemas matemáticos en los cuales no utilizaban Logo.

Dato Curioso

*La palabra Algoritmo tiene su origen en el nombre del matemático Persa "Mohamed ibn Musa **al Khwarizmi**" (825 d.C.). Su apellido fue traducido al latín como Algorismus y posteriormente paso al español como Algoritmo. Khwarizmi fue bibliotecario en la corte del califa al-Mamun y astrónomo en el observatorio de Bagdad. Sus trabajos de álgebra, aritmética y tablas astronómicas adelantaron enormemente el pensamiento matemático y fue el primero en utilizar la expresión al-yabr (de la que procede la palabra álgebra). Su trabajo con los algoritmos introdujo el método de cálculo utilizando la numeración arábiga y la notación decimal.*

En el ámbito de la computación, los Algoritmos son una herramienta que permite describir claramente un conjunto finito de instrucciones, ordenadas secuencialmente y libres de ambigüedad, que debe llevar a cabo un computador para lograr un resultado previsible. Vale la pena recordar que un programa de computador consiste de una serie de instrucciones muy precisas y escritas en un lenguaje de programación que el computador entiende (Logo, Java, Pascal, etc).

En resumen, un Algoritmo es una secuencia ordenada

de instrucciones, pasos o procesos que llevan a la solución de un determinado problema. Los hay tan sencillos y cotidianos como seguir la receta del médico, abrir una puerta, lavarse las manos, etc; hasta los que conducen a la solución de problemas muy complejos.

EJEMPLO
Un procedimiento que realizamos varias veces al día consiste en lavarnos los dientes. Veamos la forma de expresar este procedimiento como un Algoritmo:

1. Tomar la crema dental
2. Destapar la crema dental
3. Tomar el cepillo de dientes
4. Aplicar crema dental al cepillo
5. Tapar la crema dental
6. Abrir la llave del lavamanos
7. Remojar el cepillo con la crema dental
8. Cerrar la llave del lavamanos
9. Frotar los dientes con el cepillo
10. Abrir la llave del lavamanos
11. Enjuagarse la boca
12. Enjuagar el cepillo
13. Cerrar la llave del lavamanos
14. Secarse la cara y las manos con una toalla

EJEMPLO
El ejemplo de cambiar una bombilla (foco) fundida es uno de los más utilizados por su sencillez para mostrar los pasos de un Algoritmo:

1. Ubicar una escalera debajo de la bombilla fundida
2. Tomar una bombilla nueva
3. Subir por la escalera
4. Girar la bombilla fundida hacia la izquierda hasta soltarla
5. Enroscar la bombilla nueva en el plafón hasta apretarla
6. Bajar de la escalera
7. Fin

En términos generales, un Algoritmo debe ser:

- **Realizable**: El proceso algorítmico debe terminar después de una cantidad finita de pasos. Se dice que un algoritmo es inaplicable cuando se ejecuta con un conjunto de datos iniciales y el proceso resulta infinito o durante la ejecución se encuentra con un obstáculo insuperable sin arrojar un resultado.
- **Comprensible**: Debe ser claro lo que hace, de forma que quien ejecute los pasos (ser humano o máquina) sepa qué, cómo y cuándo hacerlo. Debe existir un procedimiento que determine el proceso de ejecución.
- **Preciso**: El orden de ejecución de las instrucciones debe estar perfectamente indicado. Cuando se ejecuta varias veces, con los mismos datos iniciales, el resultado debe ser el mismo siempre. La precisión implica determinismo.

Un aspecto muy importante sobre el cual los estudiantes deben reflexionar es la ambigüedad del lenguaje natural que utilizan para comunicarse diariamente con sus semejantes. La informalidad o formalidad en la comunicación depende de elementos como vocabulario, uso de comodines en lugar de vocablos precisos, uso de adverbios coloquiales en lugar de adverbios formales, etc. Es fundamental que los estudiantes aprendan a diferenciar entre comunicación informal y comunicación formal, cuya principal característica es la precisión. Los algoritmos no admiten ningún tipo de ambigüedad ya que los lenguajes de programación tienen un vocabulario restringido y preciso. Esto exige la utilización de un conjunto determinado de palabras, mandos o primitivas en cualquiera de los procedimientos que se elaboren.

ACTIVIDAD
Discutir en parejas el ejemplo de la bombilla y proponer algunas mejoras. Luego, un voluntario pasa al tablero y escribe un Algoritmo con participación de toda la clase.

Pensamiento Algorítmico

Cuando se habla de algoritmos, con frecuencia aparecen tres tipos de pensamiento que generalmente se relacionan con ellos y que se utilizan indiscriminadamente como sinónimos: Pensamiento Computacional, Pensamiento Algorítmico y Pensamiento Procedimental. Por lo tanto es importante puntualizar a qué se refiere cada uno de estos pensamientos.

Según Moursund (2006), el pensamiento computacional hace referencia a la representación y solución de problemas utilizando inteligencia humana, de máquinas o de otras formas que ayuden a resolver el problema. El pensamiento algorítmico se refiere al desarrollo y uso de algoritmos que puedan ayudar a resolver un tipo especifico de problema o a realizar un tipo especifico de tarea. Por su parte, el pensamiento procedimental se ocupa del desarrollo y utilización de procedimientos diseñados para resolver un tipo especifico de problema o para realizar un tipo especifico de tarea, pero que no necesariamente, siempre resulta exitoso.

Por otra parte y de acuerdo con un reporte del Consejo Nacional de Investigación de Estados Unidos (National Research Council, NRC, 2004), conocido como "Being Fluent with Information Technology", el Pensamiento Algorítmico incluye elementos tales como: descomposición funcional, repetición (iteración y/o recursión), organización de datos (registro, campo, arreglo, lista, etc), generalización y parametrización, diseño por descomposición de un problema en partes más pequeñas y manejables (top-down) y refinamiento.

El Pensamiento Algorítmico está fuertemente ligado al pensamiento procedimental requerido en la programación de computadores; sin embargo, su desarrollo puede conducir a los estudiantes a aproximarse guiada y disciplinadamente a los problemas de forma que este pueda transferirse a otros ambientes diferentes a los de la programación. En pocas palabras, la programación de computadores aporta al ámbito escolar un laboratorio para desarrollar habilidades

indispensables en la vida real del Siglo XXI.

Una diferencia notoria entre un algoritmo y un programa es que el algoritmo incorpora las características estructurales básicas de la computación, independientemente de los detalles de su implementación; mientras que un programa tiene un conjunto específico de detalles para resolver un problema. Se puede observar que una técnica de solución (correspondiente al algoritmo) se puede utilizar en diferentes situaciones problemáticas (correspondiente a los programas). De manera inversa, se espera que una solución exitosa de problemas incorpore procesos generales que son independientes de las situaciones específicas (NRC, 2004). Esto se conoce como experiencias de vida y los estudiantes deben adquirirlas en su paso por la educación básica y media para desempeñarse adecuadamente en su vida diaria.

Este es todo un reto para la educación, reto en el que la programación de computadores puede hacer una contribución positiva. Un programa consiste de uno o más procedimientos con instrucciones paso a paso que pueden ejecutarse en un computador; por lo tanto, utilizar el diseño de procedimientos que solucionen o ayuden a solucionar problemas con diferentes niveles de complejidad es un recurso que puede aprovechar el docente para captar el interés de los estudiantes en actividades de programación. Por ejemplo, asignar la tarea de diseñar un procesador de texto básico (ingreso del texto mediante teclado, mostrarlo en la pantalla y guardarlo en el disco duro) es una tarea relativamente sencilla. Pero el proyecto puede aumentar su complejidad si se añaden funciones para dar formato al texto (fuentes, tamaño y características especiales). Posteriormente el proyecto puede crecer si se agregan funcionalidades para manejar imágenes y tablas. Al igual que en este ejemplo, se pueden diseñar proyectos de clase interesantes para mantener motivados a los estudiantes y cuyas tareas y retos sean progresivos en complejidad; que cada nuevo reto parta de lo construido con anterioridad. En resumen, los procedimientos son un tipo particular de tarea que busca solucionar problemas específicos y al desarrollarlos se ponen en juego los pensamientos algorítmico y procedimental.

David Moursund (2006) se basó en sus propias experimentaciones y en la teoría de los cuatro estados de desarrollo cognitivo planteada por Piaget para proponer un planteamiento que amarra la computación con una escala de desarrollo cognitivo en la que se da bastante protagonismo al desarrollo del pensamiento algorítmico en los niños. Según Moursund (2006) en la etapa de las operaciones concretas los niños empiezan a manipular lógica y sistemáticamente símbolos en un computador y aprenden a apoyarse en software para resolver un rango amplio de problemas y tareas de tipo general. De esta manera, ganan habilidad considerable tanto en la utilización de lenguajes como Scratch y MicroMundos, como en la manipulación de ambientes gráficos. Posteriormente, en la etapa de operaciones formales, los estudiantes demuestran su inteligencia por medio del uso lógico de símbolos relacionados con conceptos abstractos.

Aprestamiento

Una forma motivadora y divertida de aprestamiento a la programación de computadores y que puede ayudar a los estudiantes a desarrollar los pensamientos algorítmico y procedimental consiste en que ellos realicen actividades con juegos de estrategia como “Sokoban”, “Misión Escape”, “Tetris” e “Implode”, así como ejercicios de Razonamiento Abstracto. En Sokoban se deben llevar las piedras hacia el lugar donde aparecen los prismas y para lograrlo, estas se deben empujar con el personaje teniendo cuidado en los movimientos que se hacen para no bloquear el juego ya que el personaje solo puede empujar una piedra a la vez y no puede moverlas hacia atrás, siempre hacia delante. Hay disponibles varias versiones de Sokoban para descargar y para jugar en línea.

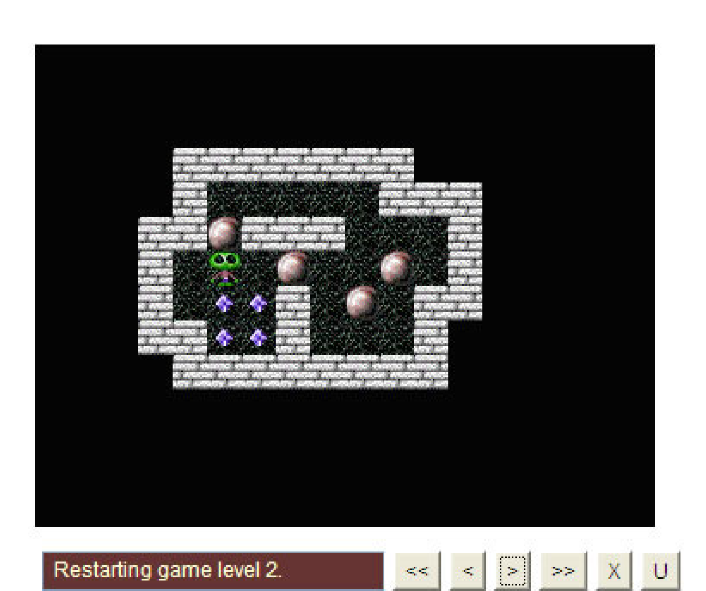


Ilustración 1: El marcianito debe mover la cuatro piedras redondas hasta ubicarlas sobre los rombos morados.
http://www.matejoven.mendoza.edu.ar/matejue/juegos/sokoban/sokoban.htm

Por su parte, el juego “Misión Escape” de la serie “Chicos del Barrio” de Cartón Networks (http://www.cartoonnetworkla.com/spanish/) se puede utilizar para mejorar la habilidad de los estudiantes para llevar a cabo tareas en forma ordenada y lógica. En este juego, los participantes deben encontrar la mejor vía de escape a través de la casa del árbol y recorrerla en la menor cantidad de movimientos posibles. Para despejar el camino de objetos hay que seguir las reglas del juego y si no se mueven los objetos precisos, en la dirección correcta y en el orden adecuado, el camino se puede bloquear.



Ilustración 2: Comienzo del nivel tres del juego "Misión Escape" de Cartoon Network. El personaje debe alcanzar la baldosa café que aparece en la parte inferior del cuadrado.

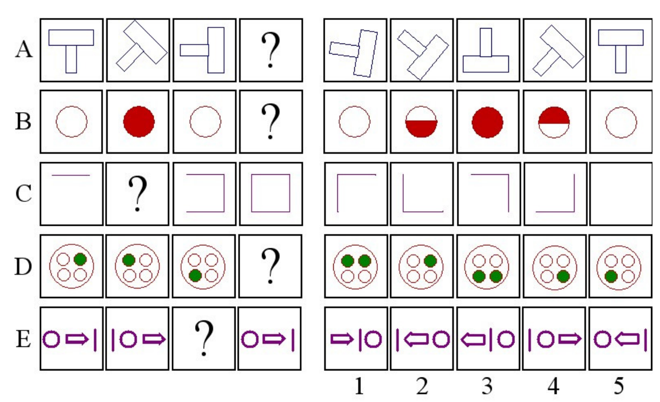


Ilustración 3: El personaje ya ha movido los obstáculos (1, 2, 3) y está a punto de alcanzar la baldosa café que le permite avanzar al nivel siguiente.

La ilustración 2 muestra el comienzo del nivel tres del juego (cada nivel es más difícil que el anterior). El personaje de "Chicos del Barrio" se encuentra en el punto de inicio y debe encontrar el mejor camino para llegar a la baldosa café de la parte inferior del cuadrado. Para lograrlo, debe mover las cajas precisas (marcadas con 1, 2 y 3), en la dirección correcta y en el orden adecuado. En la ilustración 3 se pueden apreciar las cajas movidas y el personaje a punto de alcanzar la baldosa café que le permite avanzar al nivel siguiente.

El **razonamiento abstracto** es otro tipo de actividad de aprestamiento que se puede llevar a cabo con los estudiantes para desarrollar los pensamientos algorítmico y procedimental. El razonamiento abstracto básicamente es un proceso de ordenación de objetos, situaciones o sucesos en secuencias lógicas de acuerdo con algún criterio previamente establecido. Para ello se debe comprender e interpretar los cambios en función de la forma cómo varían las características de interés de los objetos o sucesos estudiados. Todo cambio conduce a una alteración de algún aspecto del objeto, suceso o situación (Sánchez, 1993).

Actividades como la siguiente exige de los estudiantes un alto grado de observación para determinar qué es lo que cambia (figura, forma, posición, etc) y cuál es el patrón de cambio (dirección, tamaño, color, etc):



Adaptado de "Razonamiento Abstracto", Serrano (1998)

Por su parte, juegos como Guido van robot, Tetris, Implode y el mismo Sokoban, además de la versión para Computadores Personales (PCs), ofrecen versiones para los computadores OX de la iniciativa OLPC (One Laptop Per Child). Esto es importante ya que cada día más niños en América Latina disponen de estos equipos y los utilizan como herramienta para el aprendizaje.

ACTIVIDAD

Invitar a los estudiantes a reflexionar sobre el lenguaje que utiliza diariamente para comunicarse con sus padres, hermanos, profesores y compañeros. ¿Utiliza un lenguaje preciso? ¿utiliza vocablos corrientes?

ACTIVIDAD

A diferencia de los seres humanos que realizan actividades sin detenerse a pensar en los pasos que deben seguir, los computadores son muy ordenados y necesitan que el programador les especifique cada uno de los pasos necesarios y su orden lógico de ejecución.

Listar una serie de pasos para realizar una tarea y presentarlos a los estudiantes en forma desordenada para que ellos los ordenen.

Por ejemplo, ordenar los pasos para pescar:

___ El pez se traga el anzuelo.
___ Enrollar el sedal.
___ Tirar el sedal al agua.
___ Llevar el pescado a casa.
___ Quitar el Anzuelo de la boca del pescado.
___ Poner carnada al anzuelo.
___ Sacar el pescado del agua.

ACTIVIDAD

Solicitar a los estudiantes que traigan para la próxima clase los siguientes elementos:

- Arroz, lentejas o maíz (medio puñado).
- Una banda de caucho.
- Un vaso plástico.
- Un trozo de papel resistente (15cm x 15cm aproximadamente).

Divida los estudiantes en dos grupos y suministre a un grupo las siguientes instrucciones para elaborar "Maracas":

1. Recortar del papel resistente un trozo más grande que la boca del vaso plástico.
2. Introducir el arroz, las lentejas o el maíz en el vaso (cada elemento produce una sonoridad diferente).
3. Poner sobre la boca del vaso el papel.

4. Fijar el papel al vaso con ayuda de la banda de caucho.
5. Asegurarse que la boca del vaso quede sellada.

Suministre al otro grupo de estudiantes las siguientes instrucciones para elaborar “Maracas”:

1. Recortar del papel resistente un trozo más grande que la boca del vaso plástico.
2. Poner sobre la boca del vaso el papel.
3. Fijar el papel al vaso con ayuda de la banda de caucho.
4. Asegurarse que la boca del vaso quede sellada.
5. Introducir el arroz, las lentejas o el maíz en el vaso (cada elemento produce una sonoridad diferente).

Las instrucciones dadas a ambos grupos son las mismas. Sin embargo, esta actividad ilustra muy claramente la importancia que tiene el orden en que se ejecutan las instrucciones de un algoritmo.

Dato Curioso

En 1936, el lógico y matemático inglés Alan Turing (1291-1954), construyó la primera máquina conceptual como una herramienta matemática para estudiar los procesos algorítmicos. Un cálculo en una máquina de Turing consta de una secuencia de pasos que ejecuta su unidad de control. Si un problema se puede resolver en la máquina de Turing entonces es algorítmico, y recíprocamente si un problema tiene solución algorítmica, entonces se puede resolver en la máquina de Turing.
(Adaptado de ¿Qué es realmente un Algoritmo?, Escuela de Ingeniería, Colombia.)

A continuación se presentan conceptos básicos que los estudiantes deben conocer (y dominar) antes de iniciar el aprendizaje de las estructuras básicas (secuencial, decisión y repetitiva) del lenguaje algorítmico y de programación que abordaremos en la Unidad 3.

.

Los Algoritmos se puede expresar de muchas maneras, pero en esta guía se tratarán solo dos formas: Seudocódigo y Diagrama de Flujo. En **Seudocódigo** la secuencia de instrucciones se representa por medio de frases o proposiciones, mientras que en un **Diagrama de Flujo** se representa por medio de gráficos.

EJEMPLO
Elaborar un Algoritmo para calcular el área de cualquier triángulo rectángulo y presentar el resultado en pantalla.

SEUDOCÓDIGO

Paso 1: Inicio
Paso 2: Asignar el número 2 a la constante "Div"
Paso 3: Conocer la base del triángulo y guardarla en la variable "Base"
Paso 4: Conocer la altura del triángulo y guardarla en la variable "Altura"
Paso 5: Guardar en la variable "Area" el valor de multiplicar "Base" por "Altura"
Paso 6: Guardar en la variable "Area" el valor de dividir "Area" entre "Div"
Paso 7: Reportar el valor de la variable "Area"
Paso 8: Final

DIAGRAMA DE FLUJO

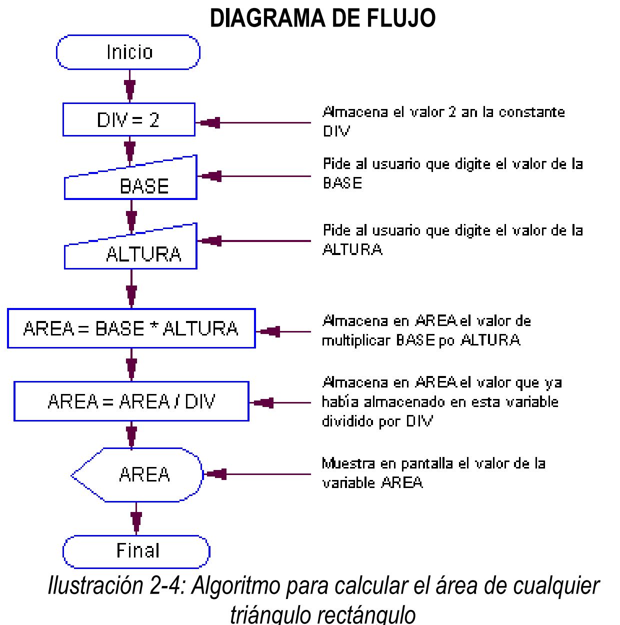


Ilustración 2-4: Algoritmo para calcular el área de cualquier triángulo rectángulo

El seudocódigo está compuesto por proposiciones informales en español que permiten expresar detalladamente las instrucciones que llevan desde un estado inicial (problema) hasta un resultados deseado (solución). Por lo regular, los algoritmos se escriben por refinamiento: se escribe una primera versión que luego se descompone en varios subproblemas (el número depende de la complejidad del problema) independientes entre sí. Si es necesario se va refinando cada vez las instrucciones hasta que las proposiciones generales en español como las del ejemplo anterior se puedan codificar en el lenguaje seleccionado para hacer la programación (en el caso de esta guía será Logo).

Utilizar Diagramas de Flujo para representar un algoritmo tiene claras ventajas, especialmente cuando son construidos por estudiantes de básica y media. Numerosas investigaciones han mostrado que el Aprendizaje Visual es uno de los mejores métodos para enseñar habilidades del pensamiento. Las técnicas que utilizan formas graficas para representar ideas e información ayudan a los estudiantes a clarificar su pensamiento, y a procesar, organizar y priorizar nueva información. Los diagramas visuales revelan patrones, interrelaciones e interdependencias además de estimular el pensamiento creativo.

La utilización de Diagramas ayuda a los estudiantes a:

- *Clarificar el pensamiento* : Ellos pueden ver cómo se conectan los procesos y se dan cuenta de cómo estos se pueden organizar o agrupar para darles el orden lógico correcto.
- *Identificar pasos erróneos* : Sobre un diagrama es más fácil identificar los cambios que se requieren para el correcto funcionamiento de un programa de computador que hacerlo sobre el código.

Los Diagramas de Flujo son una de las técnicas más utilizadas para representar gráficamente la secuencia de instrucciones de un Algoritmo. Estas instrucciones están compuestas por operaciones, decisiones lógicas y ciclos repetitivos, entre otros. La solución de un problema puede contener varios conjuntos de instrucciones (procedimientos o métodos) que tienen como finalidad ejecutar cada uno de los procesos necesarios para llegar a la solución de un problema a partir de los datos disponibles (estado inicial).

Las ventajas de diseñar un Diagrama de Flujo antes de empezar a generar el código de un programa (Rojas & Ñacato, 1980) son, entre otras:

- Forzar la identificación de todos los pasos de una solución de forma clara y lógica;
- Establecer una visión amplia y objetiva de la solución;
- Verificar si se han tenido en cuenta todas las posibilidades;
- Comprobar si hay procedimientos duplicados;
- Representar gráficamente una solución (es más simple hacerlo con gráficas que mediante palabras);
- Facilitar a otras personas la comprensión de la secuencia lógica de la solución planteada;
- Posibilitar acuerdos con base en la aproximación común a una solución de un problema, resolver ambigüedades o realizar mejoras;
- Establecer posibles modificaciones (resulta más fácil depurar un programa con el diagrama que con el listado del código);
- Agilizar la codificación (traducción) del algoritmo en un lenguaje de programación;
- Servir como elemento de documentación de la solución del problema.

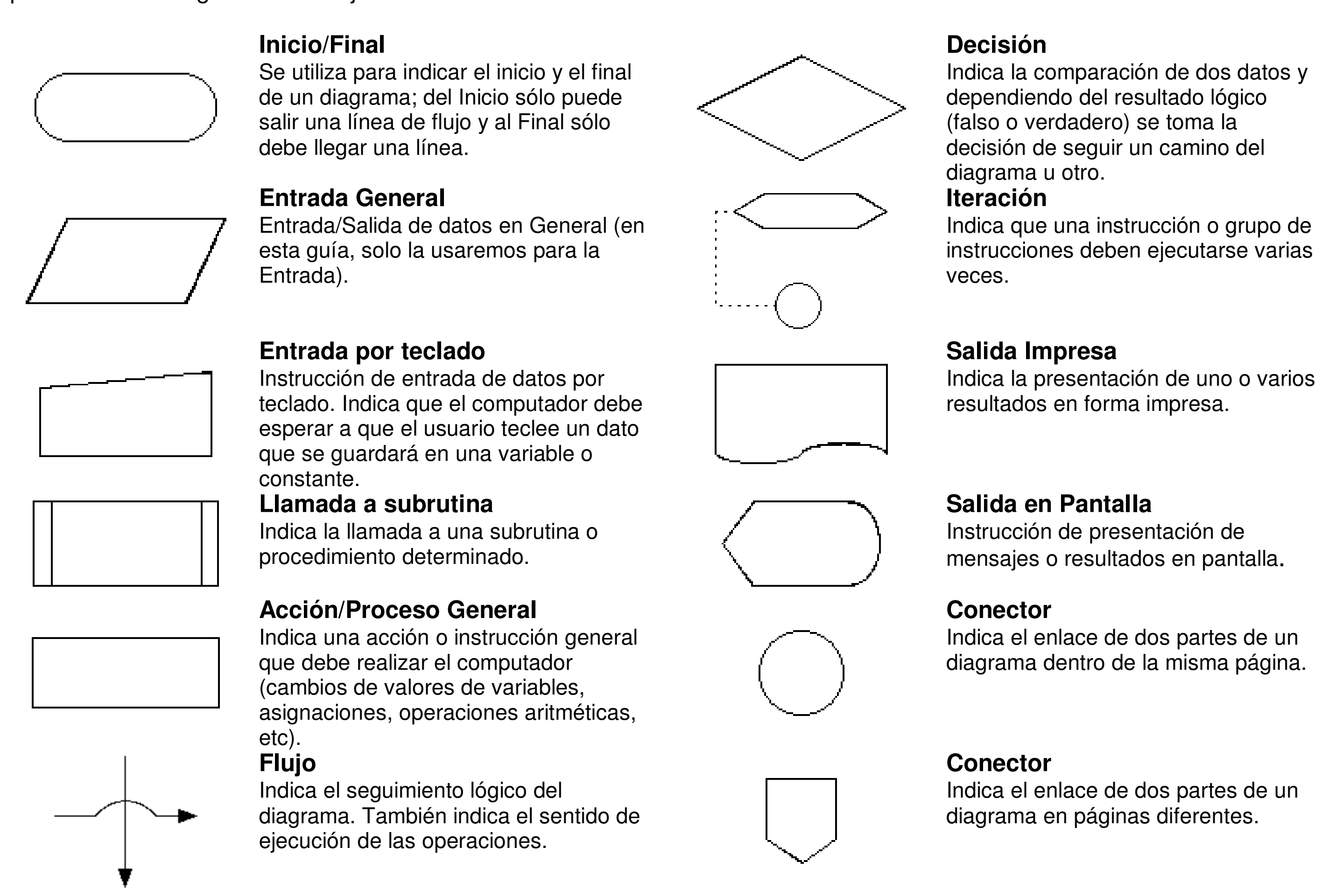
ACTIVIDAD

Basándose en la última actividad planteada en la unidad 1, elaborar un algoritmo en **seudocódigo** para cada uno de los siguientes problemas (se puede utilizar una copia de la plantilla que aparece en el anexo 7):

1. Hallar el perímetro de un cuadrado cuyo lado mide 5 cm
2. Hallar el área de un cuadrado cuyo lado mide 5 cm.
3. Hallar uno de los lados de un rectángulo cuya área es de 15 cm^2 y uno de sus lados mide 3 cm.
4. Hallar el área y el perímetro de un círculo cuyo radio mide 2 cm.
5. Hallar el área de un pentágono regular de 6 cm de lado y con 4 cm de apotema.

SIMBOLOGÍA DE LOS DIAGRAMAS DE FLUJO

La estandarización de los símbolos para la elaboración de Diagramas de Flujo tardó varios años. Con el fin de evitar la utilización de símbolos diferentes para representar procesos iguales, la Organización Internacional para la Estandarización (ISO, por su sigla en inglés) y el Instituto Nacional Americano de Estandarización (ANSI, por su sigla en inglés), estandarizaron los símbolos que mayor aceptación tenían en 1985. Los siguientes son los principales símbolos para elaborar Diagramas de Flujo:



Inicio/Final
Se utiliza para indicar el inicio y el final de un diagrama; del Inicio sólo puede salir una línea de flujo y al Final sólo debe llegar una línea.

Entrada General
Entrada/Salida de datos en General (en esta guía, solo la usaremos para la Entrada).

Entrada por teclado
Instrucción de entrada de datos por teclado. Indica que el computador debe esperar a que el usuario teclee un dato que se guardará en una variable o constante.

Llamada a subrutina
Indica la llamada a una subrutina o procedimiento determinado.

Acción/Proceso General
Indica una acción o instrucción general que debe realizar el computador (cambios de valores de variables, asignaciones, operaciones aritméticas, etc).

Flujo
Indica el seguimiento lógico del diagrama. También indica el sentido de ejecución de las operaciones.

Decisión
Indica la comparación de dos datos y dependiendo del resultado lógico (falso o verdadero) se toma la decisión de seguir un camino del diagrama u otro.

Iteración
Indica que una instrucción o grupo de instrucciones deben ejecutarse varias veces.

Salida Impresa
Indica la presentación de uno o varios resultados en forma impresa.

Salida en Pantalla
Instrucción de presentación de mensajes o resultados en pantalla.

Conector
Indica el enlace de dos partes de un diagrama dentro de la misma página.

Conector
Indica el enlace de dos partes de un diagrama en páginas diferentes.

El Diagrama de Flujo es una herramienta gráfica valiosa para la representación esquemática de la secuencia de instrucciones de un algoritmo o de los pasos de un proceso. Se recomienda consultar el siguiente componente curricular que apoya la elaboración de Diagramas de Flujo: http://www.eduteka.org/modulos.php?catx=4&idSubX=124.

REGLAS PARA LA ELABORACIÓN DE DIAGRAMAS DE FLUJO

Cuando el algoritmo se desea expresar en forma de diagrama de flujo, se deben tener en cuenta algunas reglas o principios básicos para su elaboración (Rojas & Ñacato, 1980):

- Poner un encabezado que incluya un título que identifique la función del algoritmo; el nombre del autor; y la fecha de elaboración;
- Sólo se pueden utilizar símbolos estándar (ISO 5807);
- Los diagramas se deben dibujar de arriba hacía abajo y de izquierda a derecha;
- La ejecución del programa siempre empieza en la parte superior del diagrama;
- Los símbolos de "Inicio" y "Final" deben aparecer solo una vez;
- La dirección del flujo se debe representar por medio de flechas (líneas de flujo);
- Todas las líneas de flujo deben llegar a un símbolo o a otra línea;
- Una línea de flujo recta nunca debe cruzar a otra. Cuando dos líneas de flujo se crucen, una de ellas debe incluir una línea arqueada en el sitio donde cruza a la otra (ilustración 2-5);
- Se deben inicializar las variables que se utilicen o permitir la asignación de valores mediante consulta al usuario;
- Las bifurcaciones y ciclos se deben dibujar procurando una cierta simetría;
- Cada rombo de decisión debe tener al menos dos líneas de salida (una para SI y otra para NO);
- Las acciones y decisiones se deben describir utilizando el menor numero de palabras posible; sin que resulten confusas o poco claras;
- Si el Diagrama se vuelve complejo y confuso, es mejor utilizar símbolos conectores para reducir las líneas de flujo;
- Todo el Diagrama debe ser claro, ordenado y fácil de recorrer;
- El Diagrama se debe probar recorriéndolo con datos iniciales simples (prueba de escritorio).

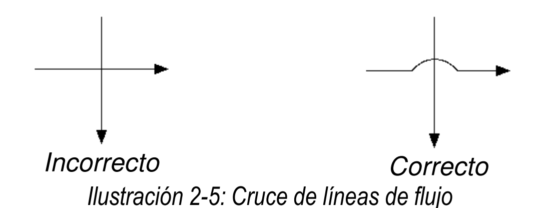


Ilustración 2-5: Cruce de líneas de flujo

Los Diagramas se pueden dibujar utilizando lápiz y papel, en cuyo caso resultan muy útiles las plantillas plásticas como la de la ilustración 2-6. Estas descargan al estudiante de la preocupación por lograr uniformidad en el dibujo.

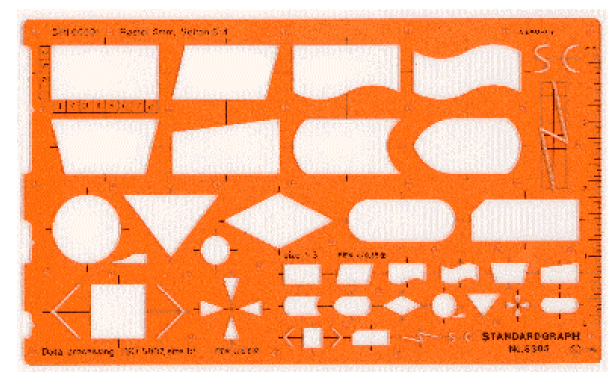


Ilustración 2-6: Plantilla StandardGraph ISO 5807 para la elaboración manual de Diagramas de Flujo.

También existe software especial para elaborar Diagramas de Flujo en forma rápida y fácil. Los programas para esta tarea permiten a los estudiantes:

- Almacenar digitalmente los diagramas construidos;
- Introducirles modificaciones fácilmente;
- Imprimir copias de los diagramas para compartirlos con compañeros o documentar sus trabajos;
- Exportarlos en varios formatos gráficos para utilizarlos en otros programas;
- Alinear y organizar los símbolos automáticamente;
- Agregar colores, tamaño de letra y sombreados para lograr una apariencia profesional;
- Ahorrar tiempo en la modificación de un diagrama ya que no es necesario hacer todo el dibujo nuevamente;

En las siguientes direcciones de Internet se puede encontrar información de software para la elaboración de Diagramas de Flujo:

- Eduteka – Diagramas de Flujo
  http://www.eduteka.org/modulos.php?catx=4&idSubX=117
- GraFl-co
  http://www.eduteka.org/pdfdir/graFlco.rar
- SmartDraw
  http://www.smartdraw.com
- WinEsquema
  http://www.softonic.com/ie/27771/WinEsquema
- Dia Win32 Installer
  http://www.softonic.com/ie/33781/dia
- DFD 1.0
  http://www.softonic.com/ie/16035/DFD
- Paraben's Flow Charter
  http://www.paraben.com/html/flow.html
- Edraw Flowchart
  http://www.edrawsoft.com/flowchart.php
- Novagraph Chartist
  http://www.tucows.com/preview/289535.html
- Flow Charting 6
  http://www.patton-patton.com
- OrgPlus
  http://www.tucows.com/preview/281861.html
- Antechinus Draw Magic
  http://www.tucows.com/preview/254904.html

ACTIVIDAD

Basándose en la actividad anterior, convertir los algoritmos elaborados en seudocódigo en **diagramas de flujo**:

1. Hallar el área de un cuadrado cuyo lado mide 5 cm.
2. Hallar uno de los lados de un rectángulo cuya área es de 15 cm^2 y uno de sus lados mide 3 cm.
3. Hallar el área y el perímetro de un círculo cuyo radio mide 2 cm.
4. Hallar el área de un pentágono regular de 6 cm de lado y con 4 cm de apotema.

Dato Curioso

En el año 1986, se introdujo en el supercomputador CRAY-2 la cifra 2^220+1, o número de Fermat 20 (que debe su nombre al matemático Pierre Fermat, 1601-1665), para averiguar si se trataba de un número primo. Al cabo de 10 días, el resultado fue "NO". Este es el cálculo realizado por un computador en el que se ha tardado más en dar una respuesta de "sí" o "no". (Libro Gunness de los Records 2002)

Para avanzar en el tema de los Algoritmos resulta indispensable que los estudiantes comprendan algunos conceptos básicos (variables, constantes, identificadores, funciones, operadores, etc), los cuales serán indispensables tanto para diseñar algoritmos como para traducirlos a un lenguaje de programación, cualquiera que este sea (Logo, Java, Visual Basic, etc).

CONCEPTOS BÁSICOS DE PROGRAMACIÓN

Variables

Para poder utilizar algoritmos con diferentes conjuntos de datos iniciales, se debe establecer una independencia clara entre los datos iniciales de un problema y la estructura de su solución. Esto se logra mediante la utilización de Variables (cantidades que se suelen denotar con letras –identificadores- y que pueden tomar cualquier valor de un intervalo de valores posibles).

En programación, las Variables son espacios de trabajo (contenedores) reservados para guardar datos (valores). El valor de una Variable puede cambiar en algún paso del Algoritmo o permanecer invariable; por lo tanto, el valor que contiene una variable es el del último dato asignado a esta. En el Algoritmo de la Ilustración 2-4, "área" es un ejemplo de Variable; en el paso 5 se guardó en ella el resultado de multiplicar "base" por "altura" y en el paso 6 se utilizó nuevamente para guardar el valor de dividir su propio contenido ("área") entre la Constante "div".

MicroMundos ofrece tres tipos de variables: Locales, Globales y de Estado. Las primeras retienen su valor el tiempo que dure la ejecución del procedimiento en el cual se utiliza. Las **variables Locales** se pueden crear con las primitivas *asigna* y *local* o en la línea del título de un procedimiento.

En MicroMundos se utiliza el comando *da* para asignar un valor a una variable o constante. La sintaxis es:

da "nombreVariable valor

que es equivalente a la forma *nombreVariable=Valor* que se utiliza en la mayoría de los lenguajes de programación.

Para utilizar el valor almacenado en una variable o constante se debe anteponer dos puntos (:) al nombre; en

:nombreVariable

los dos puntos significan "no quiero que ejecutes el comando nombreVariable; quiero el valor almacenado en nombreVariable".

Las **variables Globales** se crean con los comandos *da* o *nombra*. Estas variables solo pierden su valor cuando se cierra MicroMundos o cuando se borran con el comando bnombres.

En Scratch, se debe hacer clic en el botón "Variables" de la paleta de bloques.



Nueva variable

Luego se hace clic en el botón "Nueva variable" y se asigna un nombre a la variable, en este caso "Puntaje". Cuando se genera una variable, aparecen los bloques correspondientes a ella. Se puede escoger si la variable es para todos los Objetos (global) o solo para un Objeto (local).

Con el botón "Borrar una variable" se borran todos los bloques asociados con una variable.

Puntaje

Al hacer clic sobre el cuadrito de selección, se empieza a Informar el valor de la variable "Puntaje" en el escenario.

cambiar Puntaje por 1

Incrementa la variable en una cantidad determinada (Si se tiene más de una variable, utilice el menú desplegable para seleccionar el nombre de la variable que se desea modificar).

fijar Puntaje a 0

Inicializa la variable a un valor específico.

mostrar variable Puntaje

Muestra el monitor de la variable en el escenario.

esconder variable Puntaje

Esta opción esconde el monitor de la variable para que no aparezca en el escenario.

EJEMPLO

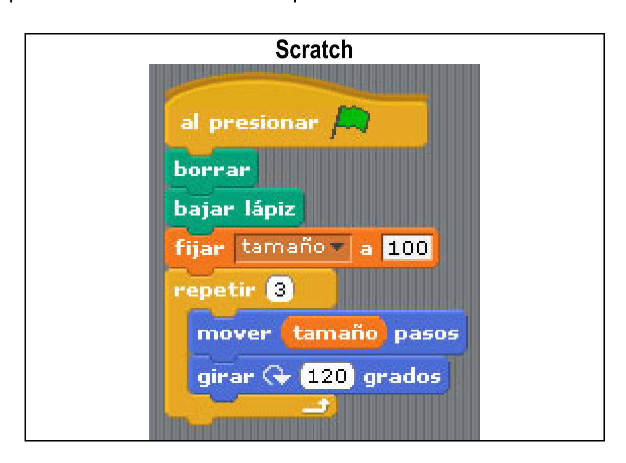
MicroMundos

para equilátero :tamaño
limpia
cp
repite 3 [adelante :tamaño derecha 120]
fin

Ahora escriba en el centro de mando de MicroMundos la palabra *equilátero* seguida por un número que representa el tamaño de cada lado del triángulo (ejemplo: *equilátero 70*).

La variable local *:tamaño* creada en la línea de título del procedimiento *equilátero*, contiene el valor 70 mientras el procedimiento se esté ejecutando. Los dos puntos *":"* que preceden el nombre de la variable *tamaño* le indican a Logo que no se quiere la palabra tamaño si no el valor que contiene la variable *tamaño*.

Scratch



En Scratch haga clic en la bandera verde para que se dibuje en el escenario un triángulo equilátero con lado 100. Para dibujar triángulos de tamaños diferentes basta con fijar la variable tamaño a otro valor.

EJEMPLO

Escriba en el área de procedimientos las siguientes líneas de código:

para tipoVariable :valorParámetro
limpia
da "variableGlobal 150
local "variableLocal
da "variableLocal 20
fin

Ahora escriba en el centro de mando de MicroMundos la secuencia de instrucciones en cursiva y debe obtener las respuestas subrayadas:

tipoVariable 70
muestra :valorParámetro
<u>valorParámetro no tiene valor</u>
muestra :variableLocal
<u>variableLocal no tiene valor</u>
muestra :variableGlobal
<u>150</u>

Observe que las variables *:valorParámetro* y *:variableLocal* no conservan su valor por fuera del procedimiento *tipoVariable*, mientras que la variable *:variableGlobal* es de tipo global y conserva su valor (150) por fuera del procedimiento donde fue creada.

En MicroMundos también existen las variables de estado que permiten conocer o modificar los componentes más importantes de una tortuga, un control o una caja de texto.

Constantes

Las Constantes se crean en Logo de la misma forma que las variables y consisten en datos que, luego de ser asignados, no cambian en ninguna instrucción del Algoritmo. Pueden contener constantes matemáticas (pi) o generadas para guardar valores fijos (3.8, "Jorge", etc). En el Algoritmo de la Ilustración 2-4, "div" es un ejemplo de Constante.

EJEMPLO

Las variables y constantes además de tener un Nombre (identificador) para poder referirnos a ellas en los procedimientos, guardan un Valor en su interior.

Nombre (identificador)	Valor
apellido	López
saldo	20000
tamaño	8.5
esTriángulo	SI

ACTIVIDAD

Pedir a los estudiantes que analicen el siguiente ejemplo y que escriban en forma de ecuación las situaciones planteadas.

Ejemplo: El doble de la edad de Carlos Andrés es 32 años:
edadCarlos es la constante donde se guarda la edad de Carlos Andrés;
R/. 2 x edadCarlos = 32

Situaciones:

1. La mitad de un valor (valor1) es 60
2. Cuatro veces un número (número1) equivale a 20
3. Un número (número2) disminuido en 5 es 18
4. El doble (elDoble) del precio de una manzana
5. La midad (laMitad) del precio de una gaseosa
6. el triple (elTriple) de mi edad

Los valores que pueden tomar valor1, número1 y

número2 (tres primeras situaciones) son constantes: 120, 5 y 23 respectivamente; no pueden tomar otros valores. Además, estas constantes son las incógnitas de las situaciones.

Los valores que pueden tomar elDoble, laMitad y elTriple son variables ya que dependen de un precio o de la edad del estudiante que resuelve el ejercicio. Los valores de estas variables hay que conocerlos para introducirlos en el problema como datos iniciales, pero no son la incógnita. Para ampliar esta actividad, el docente puede plantear nuevas situaciones o pedir a los estudiantes que planteen situaciones similares.

ACTIVIDAD

Pedir a los estudiantes que traigan tres cajas de cartón (del tamaño de las de los zapatos) y marcar cada caja con uno de los siguientes letreros: BASE, ALTURA y DIVISOR. Introducir un papel en blanco en cada una de las cajas. Solicitar a un estudiante del grupo que escriba un valor en cada uno de los papeles guardados en las cajas.

Escribir en el tablero la fórmula para calcular el área de un triángulo rectángulo:
(Base * Altura / 2)
Luego pedir a otro estudiante que escriba en el tablero los valores de los papeles guardados en cada una de las cajas y aplique la fórmula para calcular el área de un triángulo utilizando esos valores.

Repetir la operación pidiendo a otro estudiante que escriba nuevos valores en el papel de cada una de las cajas, tachando los valores anteriores.
Hacer notar que en los papeles guardados en las cajas marcadas con "BASE" y "ALTURA" se han anotado valores diferentes en cada ocasión. Este es el concepto de variable.
Hacer notar también que en el papel guardado en la caja "DIVISOR" solo se anotó un valor (2) al comienzo del ejercicio y no hubo necesidad de cambiarlo posteriormente. Este es el concepto de constante.

Esta actividad se puede adaptar para reforzar el cálculo de áreas y perímetros de otras figuras geométricas planas.

Contadores

Los contadores en MicroMundos se implementan como una estructura de programación (*da "A :A + 1*) que consistente en almacenar en una variable (*"A*) el valor de ella misma (*:A*) más un valor constante (*1*). Es muy útil para controlar el número de veces que debe ejecutarse un grupo de instrucciones.

En Scratch, se utiliza la instrucción cambiar ... por ... para incrementar la variable en una cantidad determinada.

cambiar Puntaje por 1

En este caso se almacena en la variable Puntaje el valor que ella tenga en el momento más el valor constante 1.

EJEMPLO

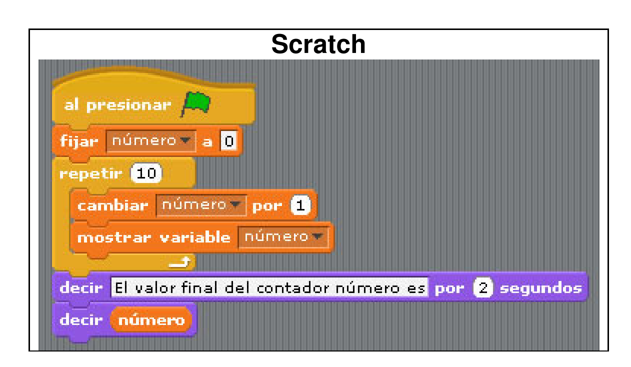
Escribir un procedimiento llamado *contador* para contar los números entre 1 y 10.

MicroMundos

para contador
bnombres
da "número 0
repite 10 [da "número :número + 1
muestra nombres]
muestra frase
[El valor final del contador número es] :número
fin

Ahora escriba en el centro de mando de MicroMundos *contador*.

Scratch



Haga clic en la bandera verde de Scratch.

Acumuladores

Estructura muy utilizada en programación (*da "A :A + :B*) y que consiste en almacenar en una variable (*"A*) el valor de ella misma (*:A*) más otro valor variable (*:B*). Es muy útil para calcular sumatorias.

EJEMPLO

Escribir un procedimiento llamado *acumulador* para calcular la sumatoria de los números entre 1 y 10.

MicroMundos

para acumulador
bnombres
da "contador 0
da "sumatoria 0
repite 10 [da "contador :contador + 1
da "sumatoria :sumatoria + :contador
muestra nombres]
muestra frase
[El valor final del acumulador sumatoria es] :sumatoria
fin

Ahora escriba en el centro de mando de MicroMundos *acumulador*.

Scratch

```
al presionar [bandera verde]
fijar contador a 0
fijar sumatoria a 0
repetir 10
    cambiar contador por 1
    cambiar sumatoria por (sumatoria + contador)
    mostrar variable contador
    mostrar variable sumatoria
decir unir [El valor final del acumulador sumatoria es =] sumatoria
```

Haga clic en la bandera verde de Scratch y el resultado debe ser 2036.

Identificadores

Los identificadores son nombres que se dan a los elementos utilizados para resolver un problema y poder diferenciar unos de otros. Al asignar nombres (identificadores) a variables, constantes y procedimientos se deben tener en cuenta algunas reglas:

- Los nombres pueden estar formados por una combinación de letras y números (*saldoMes, salario, fecha2, baseTriángulo*, etc).
- El primer carácter de un nombre debe ser una letra.
- La mayoría de los lenguajes de programación diferencian las mayúsculas de las minúsculas.
- Los nombres deben ser nemotécnicos, con solo leerlos se puede entender lo que contienen. Deben ser muy descriptivos; no utilizar abreviaturas, a menos que se justifique plenamente.
- Es conveniente utilizar una sola palabra para nombrar páginas, controles, variables, etc.
- No utilizar caracteres reservados (%, +, /, >, etc). MicroMundos admite letras acentuadas (á, é, í, ó, ú). Se debe tener en cuenta que algunos lenguajes de programación no admiten las tildes.
- No utilizar palabras reservadas por los lenguajes de programación.
- Para cumplir con convenciones ampliamente utilizadas (Jiménez, 2002), los nombres de procedimientos, variables y constantes deben empezar con minúscula. Ejemplo, *fecha*, *suma*, etc. Si es un nombre compuesto por varias palabras, cada una de las palabras (con excepción de la primera) deben empezar con mayúscula. Ejemplo: *fechaInicial*, *baseTriángulo*, etc.

El tipo de nombre –identificadores- que se asigne a variables, constantes y procedimientos es muy importante. Cuando los estudiantes dejan de trabajar en un proyecto por varios días, es más fácil para ellos retomar la actividad si los identificadores describen muy bien el contenido de variables, constantes y procedimientos. Además, el docente podrá ayudarles a revisar y depurar sus programas en forma más eficiente si estos son fáciles de leer (Feicht, 2000).

Palabras reservadas (primitivas)

Todos los lenguajes de programación definen unas palabras para nombrar sus comandos, instrucciones y funciones. Un identificador definido por el usuario no puede tener el nombre de una palabra reservada en MicroMundos.

Algunas palabras reservadas en MicroMundos	
adelante (ad)	muestra
derecha (de)	para
izquierda (iz)	limpia
atrás (at)	rumbo
repite	cp
da	sp

Las palabras reservadas no operan en Scratch ya que todas las instrucciones, incluyendo mandos y reporteros, son bloques de construcción (ver la sección Conceptos básicos de Logo en la Unidad 3). Los estudiantes no deben escribir las instrucciones, solo deben escribir los parámetros en algunas de ellas.

ACTIVIDAD

¿Cuáles de los siguientes identificadores NO son validos como nombres de Variables en MicroMundos y por qué?

1. númeroX
2. Numero X
3. 7
4. A(45+
5. VII
6. 7mesas
7. sieteMesas

En cuanto a palabras reservadas, Scratch es más flexible que MicroMundos, pues se pueden utilizar como nombres de variables aquellos identificadores que no son validos en MicroMundos: A(45+, 7, etc.

<u>TIP</u>

Es buena idea asignar, a Variables y Constantes, nombres que indiquen cuál puede ser su contenido. Por ejemplo, a una Variable que contendrá el valor de la base de un triángulo debe asignársele el nombre "baseTriangulo"; "areaCuadrado" a una que guardará el área de un cuadrado; y "radio" a una que contendrá el valor del radio de una circunferencia. Aunque MicroMundos no hace distinción entre mayúsculas y minúsculas, es buena práctica ser consistente a lo largo de todo el algoritmo en su uso (RADIOCIRC ≠ RadioCirc). Esto debido a que la mayoría de lenguajes de programación sí hacen distinción entre mayúsculas y minúsculas.

.

FUNCIONES MATEMÁTICAS

Cada lenguaje de programación tiene su conjunto de funciones matemáticas predefinidas. Estas se ejecutan haciendo referencia a su nombre. Algunas necesitan, para arrojar un resultado, que se suministre información adicional (parámetros o argumentos). Algunas de las funciones matemáticas más utilizadas en MicroMundos son:

DESCRIPCIÓN	SINTAXIS MicroMundos	Scratch
ARCO TANGENTE. Devuelve el arco tangente (la función inversa de la tangente) de su entrada. Ver tan y cos.	arctan número *Ejemplo:* *cp* *cumpleveces [i 100]* *[fx coorx + 1 fy -50 + 2 * arctan :i / 100]*	atan de 10
COSENO. Devuelve el coseno de su entrada. Ver sen y tan.	cos número *Ejemplo:* *cp* *repite 120 [fy 50 * cos 3 * coorx fx coorx + 1]*	cos de 10
EXPONENCIAL. Devuelve e a la potencia del número.	exp número *Ejemplo:* *cp* *repite 55 [fx coorx + 1 fy exp coorx / 15]*	e ^ de 10
LOGARITMO NATURAL. Devuelve el logaritmo natural (el logaritmo en base e) del número. Es el contrario de exp. Ver también log.	ln número *Ejemplo:* *muestra ln 15*	ln de 10
LOGARITMO. Devuelve el logaritmo del número. Ver ln y exp.	log número *Ejemplo:* *muestra log 15*	log de 10
PI Devuelve la constante PI.	pi *Ejemplo:* *cp* *repite 360 [ad pi * 100 / 360 de 1]* *repite 360 [ad pi * 150 / 360 iz 1]*	No disponible en Scratch
POTENCIA Devuelve el número1 elevado a la POTENCIA de número2.	potencia número1 número2 *Ejemplo:* *muestra potencia 4 2*	Scratch no tiene el operador potencia, sin embargo es fácil programarlo: http://scratch.mit.edu/projects/jualop/752239
RAÍZ CUADRADA. Devuelve la raíz cuadrada de su entrada.	rc número *Ejemplo:* *muestra rc 16*	raíz cuadrada de 10
SENO. Devuelve el seno del número en grados. Ver cos.	sen número *Ejemplo:* *cp* *repite 260 [fy 25 * sen 6 * coorx fx coorx + 1 / 2]*	sin de 6
TANGENTE. Devuelve la tangente de su entrada. Ver sen y cos.	tan número *Ejemplo:* *cp* *repite 28 [fy 8 * tan 6 * coorx fx coorx + 1 / 2]*	tan de 6

La mayoría de los lenguajes de programación disponen de una amplia variedad de datos. MicroMundos solo tiene tres tipos de datos: números, palabras y listas.

Números: se utilizan como entradas en las operaciones matemáticas. Cuando se utilizan los signos positivo (+) o negativo (-), estos deben estar pegados al número. MicroMundos acepta tanto el punto como la coma para escribir números decimales (3,14=3.14). Esto es importante tenerlo presente para no utilizar el punto para marcar la separación de miles y millones. Si asignamos a una variable el valor 20.000, MicroMundos guarda en ella el valor 20 y no 20000; si le asignamos 1.345.625 en lugar de 1345625, MicroMundos no aceptará esta notación por tener dos puntos decimales. Por su parte, Scratch solo utiliza el punto decimal; sin embargo, si usted introduce el número 6,2, Scratch lo convertirá automáticamente a 6.2.

EJEMPLO

Los siguientes son números validos en MicroMundos:

- 453
- 19,7
- 19.7
- -14,42
- 856.
- 1E6

El signo debe estar pegado al número: *muestra –3 + 6* da como resultado 3; *muestra - 3 + 6* da como resultado el mensaje "-necesita más entradas". Scratch no reconoce la notación científica: 1E6.

Palabras: Las palabras están formadas por letras y/o números. Una palabra está delimitada por espacios en blanco; sin embargo, si se quiere tener un texto conformado por dos o más palabras, este debe encerrarse entre barras (|palabra1 palabra2|).

EJEMPLO

Las siguientes son palabras validas en Logo:

- Hola
- x
- 548
- Once-Caldas
- ¿Quién?

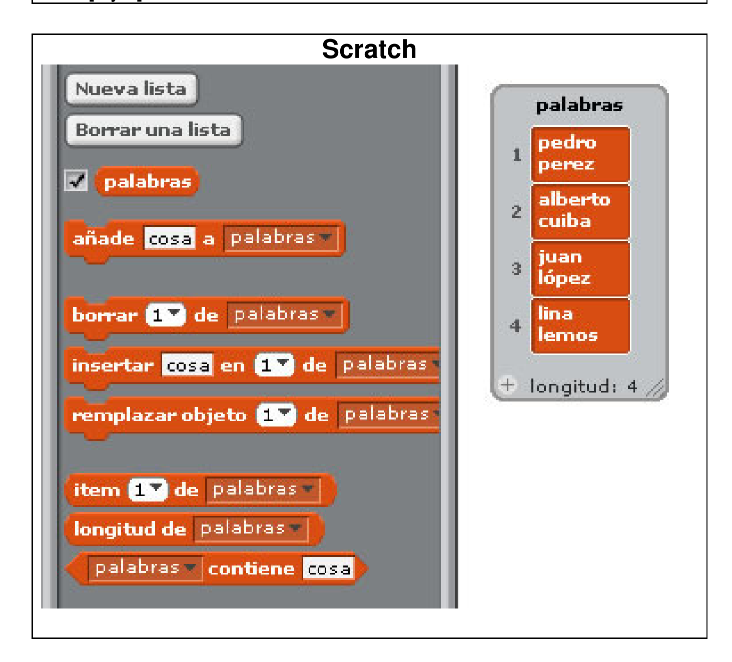
El comando *muestra "hola* da como resultado hola; *muestra "hola"* reporta hola". Varias palabras se deben tratar como una lista.

Listas: una secuencia de palabras puede manipularse igual que una sola palabra mediante el uso de listas. Una lista es una secuencia de palabras separadas por espacios en blanco y encerrada entre corchetes. Las palabras en una lista no necesitan comillas y los espacios en blanco se ignoran.

EJEMPLO

Las siguientes son listas validas en MicroMundos:

- [Esta es una lista de 7 elementos]
- [x y z]



Aunque en Scratch se pueden introducir tanto palabras como números en una variable, la operación de suma de dos variables o de elementos de una lista solo opera con números (versión 1.4). Por lo tanto no se pueden concatenar varias palabras para formar una frase con el operador +, debe utilizarse el operador "unir". Varios operadores "unir" se pueden anidar para formar una cadena de varios elementos. En el siguiente ejemplo se requieren tres espacios: uno para mostrar el nombre, otro para separar el nombre del apellido y el tercero para el apellido.

Scratch

Son símbolos que sirven para manipular datos. En MicroMundos es necesario dejar un espacio en blanco a cada lado del signo aritmético. Los operadores y las operaciones que se pueden realizar con ellos se clasifican en:

- **Aritméticos**: Posibilitan las operaciones entre datos de tipo numérico y dan como resultado otro valor de tipo numérico. Ejemplo: potencia (potencia); producto (*); división (/); suma (+); resta (-); asignación (“). Este último operador de MicroMundos presenta diferencias con el operador de asignación (=) que utilizan la mayoría de los lenguajes de programación.
- **Alfanuméricos**: Permiten operar con datos de tipo carácter o cadenas. La mayoría de los lenguajes de programación admiten el operador + para realizar la concatenación (unión) de caracteres o cadenas. Ni MicroMundos, ni Scratch tienen esta opción. En Scratch debe utilizarse, para concatenar, el operador &.
- **Relaciónales**: Permiten la comparación entre datos del mismo tipo y dan como resultado dos valores posibles: Verdadero o Falso. Ejemplo: igual a (=); menor que (<); mayor que (>).
- **Lógicos**: Posibilitan la evaluación lógica de dos expresiones de tipo lógico. Dan como resultado uno de dos valores posibles: Verdadero o Falso. Ejemplo: negación (no); conjunción (y); disyunción (o).

Orden de evaluación de los operadores

Los computadores ejecutan los operadores en un orden predeterminado. El siguiente es el orden (jerarquía) para ejecutar operadores:

1. Paréntesis (se ejecutan primero los más internos)
2. Signo (-2)
3. Potencias y Raíces (potencia y rc); Productos y Divisiones (* y /)
4. Sumas y Restas (+ y -)
5. Concatenación (+)
6. Relaciónales (=, <, >)
7. Negación (no)
8. Conjunción (y)
9. Disyunción (o)

OPERADORERS ARÍTMETICOS			
Operador	**Operación**	**Ejemplo**	**Resultado**
potencia ^	Potencia	potencia 4 2 4 ^ 2	16
*	Multiplicación	4*2	8
/	División	4/2	2
+	Suma	4+2	6
-	resta	4-2	2
“	asignación	da “A 4	Se asigna el valor de 4 a la variable A

La expresión *muestra azar 6 + 1* reporta un resultado diferente al que reporta *muestra (azar 6) + 1* ¿Por qué? Las operaciones que se encuentran entre paréntesis se evalúan primero; las que tienen el mismo orden de evaluación se ejecutan de izquierda a derecha. Los cálculos aritméticos siempre se realizan antes que cualquier otro mando Logo. Por ejemplo, en la instrucción *muestra azar 6 + 1*, la operación aritmética 6 + 1 se realiza primero que el mando Logo azar y a su vez, el mando azar se ejecuta primero que el mando muestra. En instrucciones como *muestra (azar 6) + 1* hay que tener presente que siempre se deben utilizar pares de paréntesis. En Scratch no está disponible la opción de paréntesis (hasta la versión 1.4).

ACTIVIDAD

Pedir al estudiante que escriba en el Centro de Mando (ilustración 1-2) de MicroMundos las siguientes expresiones y anote en su cuaderno las observaciones sobre los resultados de cada pareja (1 y 2; 3 y 4, 5 y 6):

1. *muestra 243 + 5 - 6 + 86 – 42*
2. *muestra 5 + 86 - 42 - 6 + 243*
3. *muestra 7 + (8 * 16)*
4. *muestra (7 + 8) * 16*
5. *muestra 24 / 4 * 8*
6. *muestra 4 * 8 / 24*

EXPRESIONES

Una Expresión está compuesta por valores, funciones, primitivas, constantes y/o variables, o por una combinación de los anteriores mediante operadores. Son Expresiones:

- Un valor (1.3, "Jorge)
- Una Constante o una Variable (divide, base, área)
- Una función (cos 60, arctan 1)
- Una combinación de valores, constantes, variables, funciones y operadores que siguen reglas de construcción y orden de evaluación de los operadores (cos 60 + 7 - :altura)

Las Expresiones pueden ser:

- **Aritméticas**: Dan como resultado un valor numérico. Contienen únicamente operadores aritméticos y datos numéricos (pi * 20 - :X)
- **Alfanuméricas**: Dan como resultado una serie o cadena de caracteres.
- **Lógicas**: Dan como resultado un valor "Verdadero" o "Falso". Contienen variables y/o constantes enlazadas con operadores lógicos (A>0 y B<=5).
- **De Asignación**: Estas Expresiones asignan el resultado de una Expresión a una Variable o a una Constante. La Expresión de Asignación (*da "área :base * :altura / 2*) asigna (*da*) el valor resultante de la Expresión Aritmética (*:base * :altura / 2*) a la variable *área*.

EJEMPLO

Para diseñar algoritmos que posteriormente puedan ser traducidos a un lenguaje de programación, es fundamental saber manejar muy bien los operadores y el orden en el que estos se ejecutan. Las fórmulas deben escribirse en una sola línea para que el computador las evalúe.

NOTACIÓN MATEMÁTICA	EXPRESIÓN
$\frac{\sqrt[2]{6^2+7}}{8^2}$	*(rc (potencia 6 2)+ 7) / (potencia 8 2)* Scratch no tiene el operador potencia, sin embargo es fácil programarlo: http://scratch.mit.edu/projects/jualop/752239
$\frac{Base * Altura}{2}$	*(base * altura / 2)* base * altura / 2

EJEMPLO

Evaluar la expresión *muestra ((potencia (5 + 3) 2) - 10) / 3 + 4 * (2 + 4)* teniendo en cuenta la jerarquía de los operadores:

R/.

*((potencia (5+3) 2) - 10) / 3 + 4 * (2 + 4)* = ((potencia 8 2) - 10) / 3 + 4 * 6
*((potencia8 2) - 10) / 3 + 4 * 6* = (64 - 10) / 3 + 4 * 6
*(64 - 10) / 3 + 4 * 6* = 54 / 3 + 4 * 6
*54 / 3 + 4 * 6* = 18 + 24
18 + 24 = 42
42

ACTIVIDAD

Tomando como modelo el ejemplo anterior y utilizando lápiz y papel, desarrollar paso a paso las siguientes expresiones. Tener en cuenta la jerarquía de los operadores:

a. (5 + 2) * (4 + 4) = 56
b. 7 + 3 * 2 + (2 - 1) = 14
c. 6 * 2 + 4 * 3 + 5 / 2 = 26,5
d. 5 + 1 * 4 / 2 * 7 - (8 + 2) = 9
e. 8 + (5 * 6) - 6 = 32
f. 9 + 5 * 3 = 24
g. 4 / 2 * (10 - 5) * 3 = 30

ACTIVIDAD

Pedir al estudiante que escriba en el Centro de Mando de MicroMundos (ilustración 1-2) las siguientes expresiones y anote en el cuaderno sus observaciones sobre el resultado:

1. *muestra 7 + 5 + 6*
2. *muestra [7 + 5 + 6]*
3. *muestra [mañana nos vemos en clase de inglés]*
4. *muestra mañana nos vemos en clase de inglés*

Obsérvese que las instrucciones 1 y 2 se diferencian en que mientras la primera da como resultado un valor numérico (18), la segunda es una lista y reporta una cadena alfanumérica de caracteres “7 + 5 + 6”.
La instrucción 4 reporta “No sé cómo hacer mañana” porque le faltan los corchetes inicial y final para que MicroMundos la considere una lista de cinco palabras.
En Scratch no son necesarios los paréntesis ya que el orden de evaluación de las expresiones es inequívoco.

EJEMPLO

Luisa Fernanda quiere llenar 5 cajas de bombones y sabe que en cada caja hay que incluir 12 bombones de menta, 14 de fresa intensa y 10 de limón. Encontrar al menos dos expresiones equivalentes para calcular el número de bombones que necesita Luisa Fernanda.

R/.

Expresión 1: **5 * 12 + 5 * 14 + 5 * 10**
¿Qué significa cada producto?
¿Qué significa la suma de los productos?

Expresión 2: **5 * (12 + 14 + 10)**
¿Qué representa la suma del paréntesis?
¿Por qué la suma se debe multiplicar por 5?
¿Las dos expresiones dan el mismo resultado?
Ejercicio adaptado de “Cuenta Jugando 5”; Página 48 (Casasbuenas & Cifuentes, 1998b).

ACTIVIDAD

da “númeroA 5 (asigna el valor 5 a la Constante númeroA)
da “númeroB 8 (asigna el valor 8 a la Constante númeroB)
Utilizando la información de los valores asignados a las Constantes númeroA y númeroB, evaluar las siguientes expresiones:

1. 20 + :númeroA
2. :númeroA + 3 * :númeroB
3. :númeroA > :númeroB
4. :númeroA + "123
5. 4 + :númeroA - :númeroB

UNIDAD 3: ESTRUCTURAS BÁSICAS

LAS ESTRUCTURAS

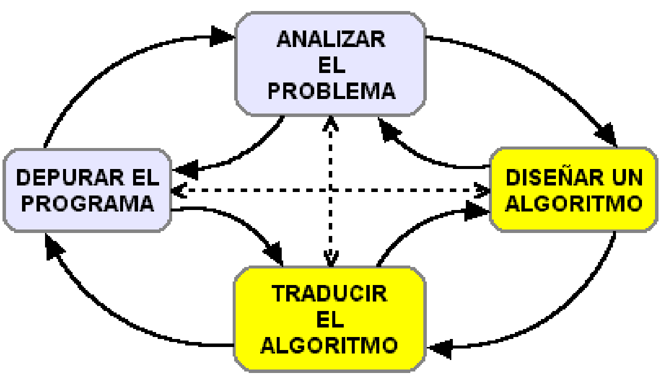


Ilustración 3-1: Fases segunda y tercera del ciclo de programación.

Un Algoritmo está compuesto por instrucciones de diferentes tipos, organizadas secuencialmente, en forma de estructuras de control. De estas estructuras, las más comunes y que se cubren en esta guía son las siguientes:

- Secuencial.
- Iterativa (repetición).
- Condicional (decisión, selección).

Una estructura de control se define como un esquema que permite representar ideas de manera simplificada y que bajo condiciones normales, es constante (Trejos, 1999).

El uso del diseño descendente en los programas, la ejecución de operaciones secuenciales, la utilización de ciclos repetitivos y, la toma de decisiones y alternativas de proceso, ofrecen amplias posibilidades para resolver problemas mediante la construcción de procedimientos (Castellanos & Ferreyra, 2000b).

Un famoso teorema de los años sesenta, formulado por Edsger Wybe Dijkstra, demostraba que se puede escribir cualquier programa utilizando únicamente la tres estructuras de control mencionadas. Actualmente, la programación orientada a objetos (POO) busca reducir al máximo la cantidad de estructuras de control mediante el uso de polimorfismo; pero aún así, todavía se vale de estas estructuras para construir métodos, los cuales podría decirse que son equivalentes a los procedimientos que se plantean en esta guía.

Para traducir los algoritmos diseñados a un lenguaje de programación que el computador pueda entender, en esta guía se utilizan dos entornos de programación basados en Logo: MicroMundos y Scratch. Los docentes interesados en conocer estos ambientes de programación, pueden descargar gratuitamente las correspondientes Guías de Referencia:

- MicroMundos (proyecto Teddi - PDF; 560KB)
  http://www.eduteka.org/pdfdir/ManualMicroMundos.pdf
- Scratch (MIT - PDF; 1.5MB)
  http://www.eduteka.org/pdfdir/ScratchGuiaReferencia.pdf

TIP

Edsger Wybe Dijkstra nació en Rotterdam, (Holanda) en 1930. En 1956 anunció su algoritmo de caminos mínimos; posteriormente propuso el algoritmo del árbol generador minimal. A principios de la década de los 60, aplicó la idea de exclusión mutua a la comunicación entre un computador y su teclado. Su solución de exclusión mutua ha sido usada en muchos procesadores y tarjetas de memoria desde 1964, año en el que fue utilizada por IBM en la arquitectura del "IBM 360". El siguiente problema del que se ocupó Dijkstra fue el de los filósofos comensales. En este problema, cinco filósofos están sentados en una mesa circular con un plato de arroz delante y un palillo a cada lado, de manera que hay cinco palillos en total. El problema trata sobre el uso de recursos comunes sin que los procesos (los filósofos) lleguen a una situación de bloqueo mutuo; además, que los recursos se utilicen por todos los procesos de la manera más eficiente. Dijkstra también contribuyó a desterrar el comando GOTO de la programación: el comando "GOTO es considerado dañino. Cuantas más sentencias GOTO tenga un programa, más confuso será el código fuente".

Logo

Logo es un lenguaje de programación con un número limitado de palabras y de reglas gramaticales si se lo compara con lenguajes humanos, como el castellano o el inglés. Las instrucciones en Logo son equivalentes a las oraciones en castellano y las reglas para construir esas instrucciones son más simples que las reglas gramaticales de nuestro idioma, pero mucho más precisas.

Logo fue desarrollado por Seymour Paper en el MIT en 1968 con el fin de utilizarlo en el ámbito educativo. Con este lenguaje de programación, inicialmente los niños dan instrucciones a una tortuga (adelante, atrás, derecha, izquierda, etc) para elaborar dibujos sencillos. Pero a medida que logran dominio del lenguaje, ellos utilizan comandos más sofisticados para realizar creaciones más complejas.

Desde 1968 a la fecha se han creado numerosas versiones de Logo (http://www.elica.net/download/papers/LogoTreeProject.pdf); sin embargo, en esta Guía se utilizarán los entornos de programación basados en Logo "MicroMundos" y "Scratch". Ambos ofrecen una serie de comandos llamados "primitivas" para construir procedimientos (ver una lista básica de primitivas en el Anexo 1).

Para obtener mayor información sobre Logo, se recomienda visitar el sitio Web de Daniel Ajoy http://neoparaiso.com/logo/

MicroMundos

MicroMundos (http://www.micromundos.com) es un entorno de programación desarrollado por la compañía canadiense LCSI. La lista de primitivas es más extensa que la de Scratch; la cual, junto a los procedimientos elaborados por el programador, se pueden clasificar en dos categorías: Mandos y Reporteros.

Los **Mandos** hacen algo. Por ejemplo, *derecha* y *muestra* son Mandos (*derecha 120*, *muestra rumbo*, etc). Los **Reporteros** informan sobre el resultado de un cálculo o sobre el estado de un objeto. Por ejemplo, *rumbo* y *primero* son Reporteros. En la instrucción *muestra rumbo*, el Reportero *rumbo* devuelve al Mando *muestra* el rumbo actual de la tortuga, *rumbo* como tal no puede hacer nada.

Se requiere que los estudiantes aprendan a interpretar las instrucciones tal como lo hacen con las oraciones en castellano (sustantivos, verbos, conectores, adjetivos, etc). En MicroMundos la primera palabra de una instrucción siempre debe ser un Mando, por lo tanto, los Reporteros solo se pueden utilizar como entradas de un Mando o de un procedimiento.

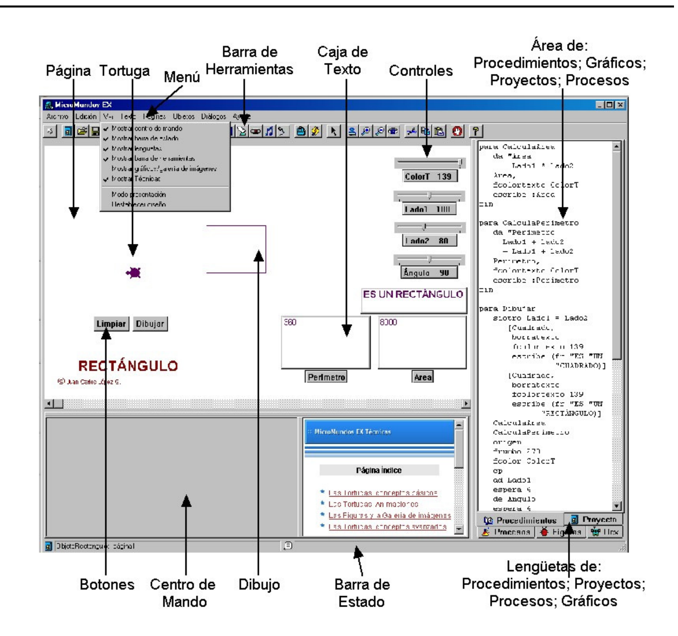


Ilustración 3-2(a): Área de trabajo de MicroMundos (interfaz del programa)

EJEMPLO 3-1

Digitar en el Centro de Mando (Ilustración 3-2) la instrucción:
rumbo
MicroMundos devuelve un mensaje de error:
No sé qué hacer con 0
Digitar la siguiente instrucción en el Centro de Mando:
muestra rumbo
MicroMundos devuelve el rumbo actual de la tortuga:
0
En este caso el resultado de *rumbo* es reportado a *muestra*.

Los docentes interesados en conocer más a fondo MicroMundos, pueden descargar una versión de prueba por 30 días del sitio: http://www.micromundos.com Además, pueden descargar gratuitamente la Guía de Referencia en español: Proyecto Teddi; PDF; 560KB) http://www.eduteka.org/pdfdir/ManualMicroMundos.pdf

Aunque MicroMundos es bueno y ampliamente utilizado en escuelas de América Latina, su costo puede constituir una restricción para muchas Instituciones Educativas. Sin embargo, entre las alternativas gratuitas (licencia GNU) de Logo hay una que utiliza en núcleo de Logo creado por Brian Harvey de la Universidad de Berkeley: Microsoft Windows Logo (MSWLogo). Si bien, no es un ambiente de programación tan atractivo y elaborado como MicroMundos, esta versión de Logo es gratuita y tiene una traducción al español realizada por Javier López-Escobar que se puede descargar de: http://sourceforge.net/projects/mswlogoes Sin embargo, desde 2006, y debido al notorio abandono de MSWLogo por parte de su desarrollador, David Costanzo asumió la continuación del desarrollo de MSWLogo con el nombre FMSLogo: http://fmslogo.sourceforge.net/

Scratch

Scratch (http://scratch.mit.edu/) es un entorno de programación desarrollado recientemente por un grupo de investigadores del Lifelong Kindergarten Group del Laboratorio de Medios del MIT, bajo la dirección del Dr. Michael Resnick.

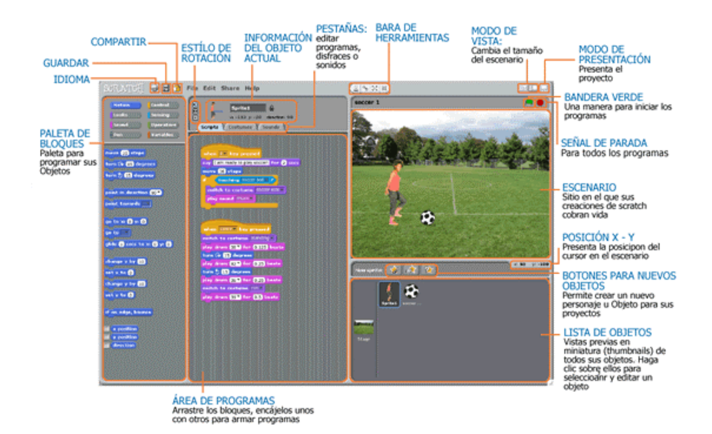


Ilustración 3-2(b): Área de trabajo de Scratch
(Ver imagen con mayor tamaño en el Anexo 8)

Aunque Scratch es un proyecto de código abierto, su desarrollo es cerrado. El código fuente se ofrece de manera libre y gratuita. Actualmente hay disponibles versiones oficiales para Windows, Mac y Sugar (XO); Además, hay versiones no oficiales para Linux.

Este entorno de programación aprovecha los avances en diseño de interfaces para hacer que la programación sea más atractiva y accesible para todo aquel que se enfrente por primera vez a aprender a programar. Según sus creadores, fue diseñado como medio de expresión para ayudar a niños y jóvenes a expresar sus ideas de forma creativa, al tiempo que desarrollan habilidades de pensamiento algorítmico y de aprendizaje del Siglo XXI, a medida que sus maestros superan modelos de educación tradicional en los que utilizan las TIC simplemente para reproducir prácticas educativas obsoletas.

Entre las características más atractivas de Scratch, adicionales a las mencionadas en el párrafo anterior, que lo hacen interesante para muchas Instituciones Educativas se cuentan: es gratuito, tiene el respaldo del MIT, está en permanente desarrollo (cada año se liberan aproximadamente dos versiones con cambios significativos), corre en computadores con bajas prestaciones técnicas y se puede ejecutar desde una memoria USB (pen drive), entre otras.

Scratch es una muy buena alternativa a MicroMundos; sin embargo, las Instituciones Educativas que ya cuentan con MicroMundos, pueden utilizar ambos entornos ya que puede resultar benéfico para los estudiantes exponerlos a diferentes ambientes de programación en los que puedan realizar las mismas cosas.

Los docentes interesados en conocer más a fondo el ambiente de programación Scratch, puede descargarlo gratuitamente de la siguiente dirección: http://scratch.mit.edu/. Además, pueden descargar la Guía de Referencia en español (PDF; 1.5MB):
http://www.eduteka.org/pdfdir/ScratchGuiaReferencia.pdf

Fundamentos de programación

Una vez descritos brevemente los entornos de programación que se utilizan a lo largo de esta Guía, se tratarán a continuación una serie de conceptos básicos requeridos para empezar a utilizar dichos entornos.

COMENTARIOS

Los comentarios no tienen ningún efecto en la ejecución del algoritmo. Se utilizan para aclarar instrucciones que puedan prestarse a confusión o como ayuda a otras personas que deben leerlo y entenderlo. La mayoría de los lenguajes de programación ofrecen la posibilidad de comentar el código de los programas.

Los comentarios en un procedimiento de MicroMundos se hacen con el punto y coma (;). MicroMundos ignora el texto entre el punto y coma (;) y el final de la línea.

MicroMundos

```
para lectura
  local "valorUno      ; declarar variables y constantes
  local "ValorDos
  pregunta [Ingrese el primer valor ]    ; ingresar  valorUno
  da "valorUno respuesta
  pregunta [Ingrese el segundo valor ] ; ingresar  valorDos
  da "valorDos respuesta
  muestra frase [El primer valor es  ]   :valorUno
  muestra frase [El segundo valor  es  ]  :valorDos
fin
```

En Scratch, los comentarios se agregan en una caja de texto amarilla que se crea al hacer clic derecho sobre cualquier parte del área de programas (zona central gris) y seleccionar la opción "añadir comentario".

Scratch

```
al presionar [bandera verde]
preguntar [Ingrese el primer valor...] y esperar
fijar [valorUno] a (respuesta)
preguntar [Ingrese el segundo valor...] y esperar
fijar [valorDos] a (respuesta)
decir (unir [El primer valor es = ] (valorUno)) por (2) segundos
decir (unir [El segundo valor es = ] (valorDos)) por (2) segundos
```

este procedimiento solicita dos valores y luego los muestra

Los algoritmos diseñados como ejemplos en esta Guía no están optimizados ya que con ellos se busca mostrar explicita y detalladamente las operaciones básicas requeridas para una solución y no la forma óptima de solución. Algunos ejemplos exponen de manera

deliberada la forma larga y poco elaborada, para luego presentar la forma optimizada.

EJEMPLO 3-2
Escribir un procedimiento que se llame triangulo para hallar el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm. Introducir en el código comentarios que aclaren lo que está sucediendo en cada uno de los pasos importantes.
R/

ANÁLISIS DEL PROBLEMA

Formular el problema: Ya se encuentra claramente planteado.
Resultados esperados: El área de un triángulo rectángulo.
Datos disponibles: Base, Altura, Hipotenusa, tipo de triángulo. La incógnita es el área y todos los valores son constantes. El valor de la hipotenusa se puede omitir. El estudiante debe preguntarse si sus conocimientos actuales de matemáticas le permiten resolver este problema; de no ser así, debe plantear una estrategia para obtener los conocimientos requeridos.
Restricciones: Utilizar las medidas dadas.
Procesos necesarios: Guardar en dos variables (BASE y ALTURA) los valores de Base y Altura; Guardar en una constante (DIV) el divisor 2; aplicar la fórmula BASE*ALTURA/DIV y guardar el resultado en la variable AREA; comunicar el resultado (AREA).

DISEÑO DEL ALGORITMO

Inicio
DIV = 2
BASE=3
ALTURA=4
Asigna valores a las constantes DIV, BASE y ALTURA
AREA = BASE * ALTURA / DIV
Calcula el área y la almacena en la variable AREA
AREA
Imprime el valor almacenado en la variable AREA
Final

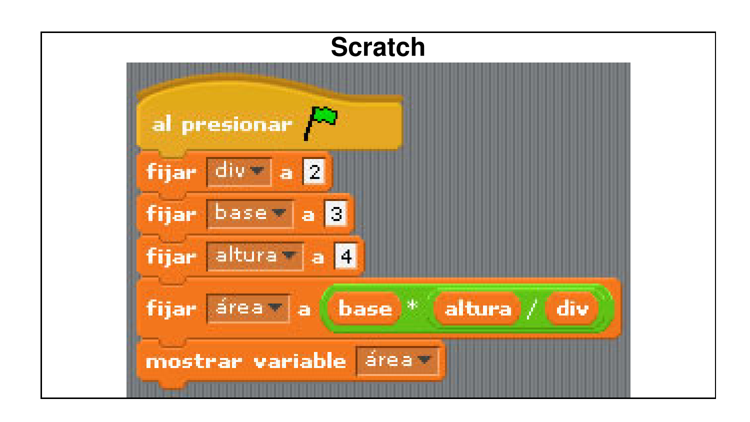
TRADUCCIÓN DEL ALGORITMO

MicroMundos

```
para triángulo
  local "div       ;declara las constantes como locales.
  local "base
  local "altura
  local "área      ;declara esta variable como local.
  da "div 2        ;almacena 2 en la constante div (div=2)
  da "base 3       ;equivalente a base=3
  da "altura 4
  da "área :base * :altura / :div     ;aplica la fórmula.
  muestra :área     ;reporta el contenido de la variable área
fin
```

Tal como se puede comprobar al ejecutar el procedimiento anterior, el texto entre un punto y coma y el final de una línea de código no produce ningún resultado, ni causa errores de sintaxis.

Scratch



PROCESOS
Se llama procesos a todas las instrucciones contenidas en un algoritmo para:

- declarar variables y constantes (solo en MicroMundos);
- asignar valores iniciales a variables y constantes;
- leer datos que suministra el usuario por medio del teclado o del ratón (mouse);
- realizar operaciones matemáticas (aplicar fórmulas);
- reportar o mostrar contenidos de variables y constantes;
- mostrar en pantalla resultados de procedimientos activados por el programa.

Vale la pena recordar que la declaración de variables y constantes se realiza en MicroMundos de la siguiente forma:
local "nombreVariable (Ej. *local "altura*).

La sintaxis para asignación de un valor a una variable o constante es:
da "nombreVariable Valor (Ej. *da "altura 4*).

Luego de asociar valores a variables o constantes, estas se pueden utilizar anteponiendo dos puntos (:) al nombre de la variable o constante:
da "nuevaAltura 2 + :altura

El Mando *da* asigna a la variable *nuevaAltura* el valor 2 más el contenido de la variable *altura*.

Por otra parte, al momento de crear una variable en Scratch, el entorno pregunta si esta será visible "para todos los objetos" (global) o solo será visible "para este objeto" (local).

Scratch



INTERACTIVIDAD

La interactividad entre el usuario y el programa es un recurso muy valioso en programación y se logra permitiendo la comunicación con el programa mediante la utilización del teclado y/o el ratón (mouse).

En MicroMundos, la forma más común de interactuar con un programa es por medio de controles y botones, los cuales se manipulan con el ratón. Ambos se pueden programar para que realicen acciones (ejecutar un procedimiento, ir a otra página, cambiar el valor a una variable, etc).

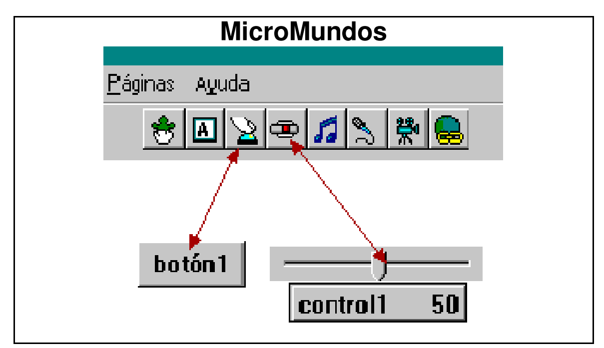
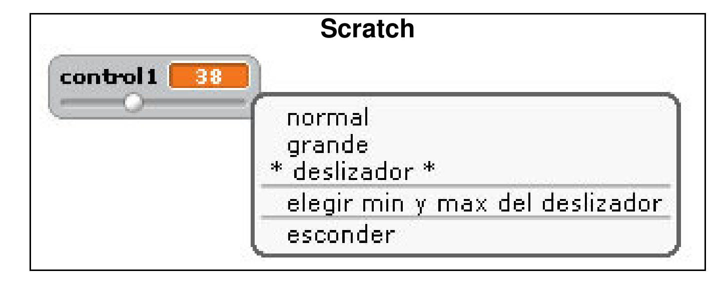


Ilustración 3-3: Con el ratón se puede hacer clic sobre los Botones y deslizar el indicador de los Controles (a la izquierda disminuye y hacia la derecha aumenta).

En Scratch algunos objetos se les puede dar forma de botón y programarlos para que cumplan la misma función que cumplen estos en MicroMundos. Respecto a los controles, en Scratch cada variable se puede convertir en un control deslizante. Basta con hacer clic derecho sobre cualquier variable que se muestre en el escenario y seleccionar "deslizador"; luego se hace clic derecho nuevamente y se eligen los valores mínimo y máximo que puede almacenar esa variable mediante la manipulación con el deslizador.



Otra forma de interactuar con un programa consiste en que el usuario utilice el teclado para responder a las preguntas que le plantea el programa. En MicroMundos las respuestas que el usuario aporta se pueden almacenar en las variables correspondientes con el comando *respuesta*. Para mostrar lo que el usuario contesta se puede utilizar varios comandos de MicroMundos (*mostrar, escribe, anuncia, anuncia frase*).

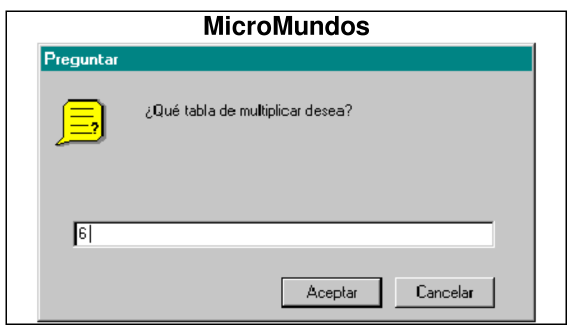


Ilustración 3-4: Los comandos pregunta y respuesta permiten interactuar con el usuario para obtener información de él.

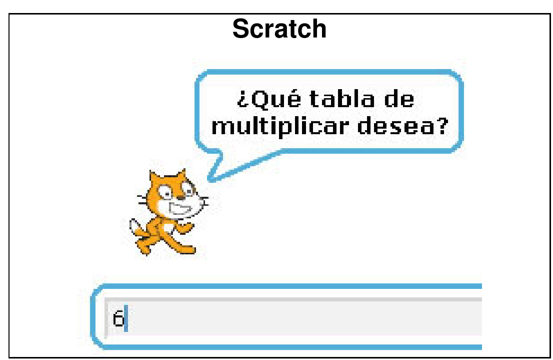


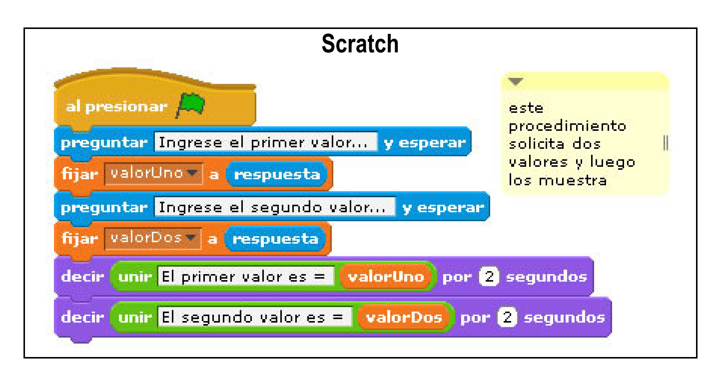
Ilustración 3-5: Los comandos pregunta y respuesta permiten interactuar con el usuario para obtener información de él.

En Scratch las respuestas que el usuario aporta se pueden almacenar en variables con el comando respuesta. Para mostrar lo que el usuario contesta se puede utilizar los comandos "decir" y "pensar".

EJEMPLO 3-3

MicroMundos

```
para lectura
  local "valorUno     ; declarar variables y constantes
  local "ValorDos
  pregunta [Ingrese el primer valor ]   ; ingresar  valorUno
  da "valorUno respuesta
  pregunta [Ingrese el segundo valor ] ; ingresar  valorDos
  da "valorDos respuesta
  muestra frase [El primer valor es  ]  :valorUno
  muestra frase [El segundo valor  es  ] :valorDos
fin
```



El ejemplo 3-6 de la sección "Estructura Secuencial" ilustra muy bien la interactividad que se puede establecer entre usuario y programa. La interactividad

facilita generalizar la solución de un problema. Por ejemplo, si se elabora un procedimiento que calcule el área de un triángulo rectángulo, cada vez que se cambien los valores iniciales de "base" y "altura", estos deben actualizarse en el cuerpo del procedimiento. La interactividad, por el contrario, permite que cada vez que se ejecute el procedimiento, este le pregunte al usuario por los valores de "base" y "altura", y los tome para calcular el área sin la necesidad de modificar nada en el procedimiento.

PROCEDIMIENTOS

Según Papert (1993), los problemas que experimentan muchos estudiantes con las matemáticas se deben más a la falta de comprensión de los algoritmos apropiados, que a la falta de manejo de los conceptos involucrados en estos. Cuando experimentan problemas con la suma, lo primero que se debe revisar es el procedimiento de la adición.

Los procedimientos se utilizan muy a menudo en la vida diaria. Participar en un juego o dar instrucciones a alguien que se encuentra perdido son actividades que requieren pensamiento procedimental o algorítmico. Los estudiantes activan a diario este tipo de pensamiento sin percatarse y sin hacer algún tipo de reflexión sobre esto. De hecho, ellos disponen de un conocimiento procedimental que utilizan en muchos aspectos de sus vidas, tanto para planear una estrategia en un juego, como para dar instrucciones a alguien perdido en el vecindario. Lo curioso es que en raras ocasiones utilizan este conocimiento procedimental en las clases de matemáticas. Los ambientes Logo ayudan a los estudiantes a hacer conciencia de la idea de procedimiento, ya que en él, los procedimientos se convierten en algo que se puede nombrar, manipular y reconocer (Papert, 1993).

Además del pensamiento algorítmico, los razonamientos temporal y condicional juegan un papel muy importante en la gestión, organización y planificación de cualquier procedimiento. Estos facilitan al programador plantear y seguir instrucciones, fragmentar una tarea en módulos con funciones precisas y plantear decisiones que un procedimiento debe tomar de acuerdo a ciertas condiciones. La programación esta muy ligada al control de la sucesión temporal de instrucciones organizadas en secuencias. Este control, guiado por un razonamiento temporal, es necesario para organizar el orden de las instrucciones a ejecutar, para llamar otros procedimientos en el orden correcto y ejecutar ciertas instrucciones mientras una condición se dé o a partir de cuando esta se dé (Dufoyer, 1991). Y, según Friedman (1982) y Piaget (1946), citados por Dufoyer (1991), hay que esperar necesariamente hasta los 7 u 8 años (nivel de las operaciones concretas) para que estas operaciones sean posibles. Por lo tanto, utilizar la programación de computadores con estudiantes de cuarto grado en adelante es viable si se hace con el objetivo de ofrecerles oportunidades de acceso a conceptos relacionados con procedimientos, creando condiciones para que ellos ejerciten en forma efectiva y divertida el pensamiento algorítmico y los razonamientos temporal y condicional.

En este sentido, la utilización del área de procedimientos de MicroMundos (ilustración 3-2) ofrece mayores posibilidades para gestionar, organizar y planificar, en contraposición a introducir instrucciones una a una en el "centro de mando" (lo que permite al estudiante ver inmediatamente cuál es el efecto que produce cada instrucción ejecutada).

Como se expuso en la Unidad 1, los procedimientos son módulos con instrucciones que inician con el comando "*para*" y que el computador ejecuta automáticamente, una tras otra, hasta encontrar el comando "*fin*". Todo procedimiento debe tener un nombre que lo identifique y que sirve para ejecutarlo cuando se ejecuta dicho nombre. Pero antes de ejecutar un procedimiento, los estudiantes deben utilizar pensamiento algorítmico, razonamiento temporal y razonamiento condicional, para determinar y escribir todas las instrucciones que lo deben componer y el orden lógico de ejecución.

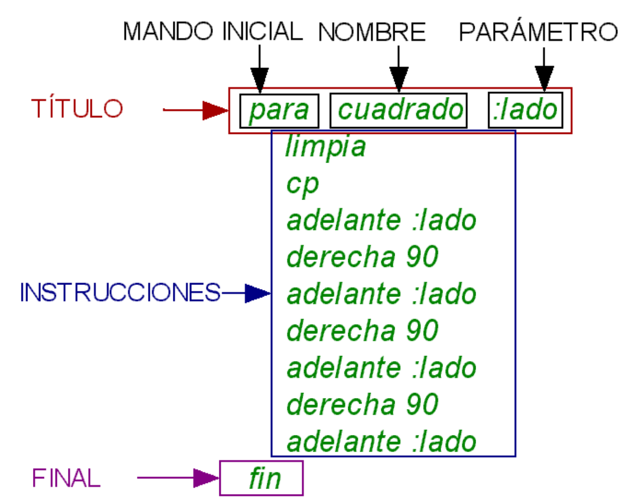


Ilustración 3-5: Elementos que componen el procedimiento "cuadrado"; el parámetro es el único elemento opcional.

Por otra parte, el vocabulario disponible en cualquier proyecto Logo está compuesto por los comandos propios del lenguaje (Mandos y Reporteros) más los nombres de los procedimientos definidos en ese proyecto. En otras palabras, los procedimientos definidos (en este caso "cuadrado") entran a formar parte del vocabulario de ese proyecto. Cuando se abre un proyecto nuevo, esos procedimientos ya no están disponibles; por tanto, para utilizarlos hay que copiarlos de un proyecto existente y pegarlos en el nuevo proyecto.

Todo procedimiento debe tener una línea de título que incluye un nombre; el cual, al invocarlo, ejecuta en orden, una a una, las líneas de instrucciones que contiene hasta que llega a la línea con el Mando *fin*.

Para escribir procedimientos se deben tener en cuenta las siguientes recomendaciones:

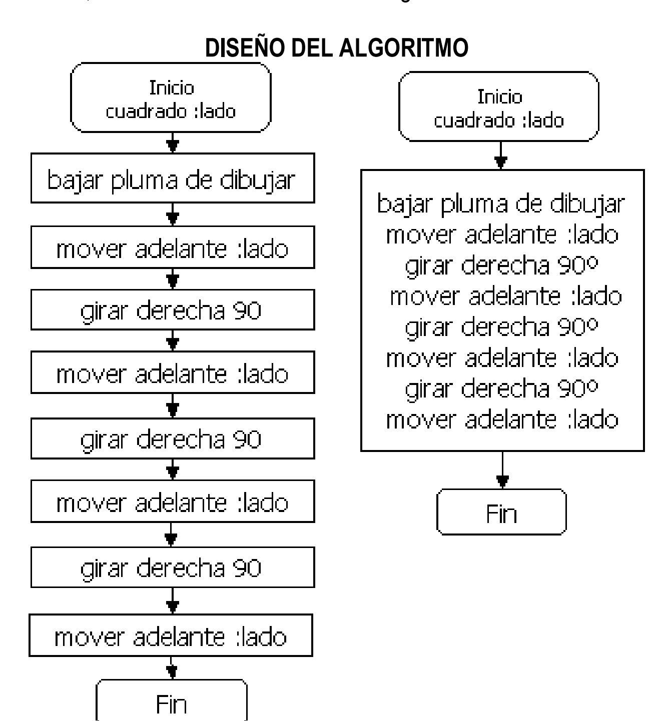
- El título está compuesto por: el Mando "*para*", el nombre del procedimiento y los parámetros (opcional).
- El nombre del procedimiento debe ser una palabra. Cuando se deben utilizar varias palabras para nombrar un procedimiento, se escriben sin dejar espacio entre ellas y con mayúscula la primera letra de la segunda palabra y de las subsiguientes palabras (ejemplo: *valorMasAlto*).
- Cada instrucción completa debe ocupar una línea de código
- Puede contener líneas en blanco
- En la última línea solo puede aparecer el Mando "*fin*"
- Desde un procedimiento se pueden invocar otros procedimientos.

Ejemplo 3-4
Escribir un procedimiento para dibujar en la pantalla un cuadrado de tamaño variable.
R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Ya está claramente planteado.
Resultados esperados: El dibujo de un cuadrado en la pantalla.
Datos disponibles: El tamaño de los lados del cuadrado debe ingresarlo el usuario; se sabe que todos los ángulos internos de un cuadrado son de 90 grados. El estudiante debe preguntarse si sus conocimientos actuales de matemáticas le permiten resolver este problema; de no ser así, debe plantear una estrategia para obtener los conocimientos requeridos.
Restricciones: El tamaño del cuadrado lo suministra el usuario.
Procesos necesarios: Leer el tamaño del cuadrado como un parámetro llamado *lado*; bajar la pluma de dibujar; avanzar adelante una distancia igual a lado; girar 90 grados a la derecha; avanzar adelante una distancia igual a lado; girar 90 grados a la derecha; avanzar adelante una distancia igual a lado; girar 90 grados a la derecha; avanzar adelante una distancia igual a lado.

DISEÑO DEL ALGORITMO



Aquí tenemos dos formas equivalentes e igualmente correctas de algoritmo para dibujar un cuadrado. El algoritmo de la izquierda utiliza un rectángulo de proceso para cada instrucción; el de la derecha agrupa en un solo rectángulo las instrucciones de proceso adyacentes.

TRADUCCIÓN DEL ALGORITMO

El siguiente procedimiento es la traducción al lenguaje Logo del algoritmo representado por el diagrama de flujo. Pedir a los estudiantes que lo escriban en el área de procedimientos.

MicroMundos
```
para cuadrado :lado
   limpia
   cp
   adelante :lado
   derecha 90
   adelante :lado
   derecha 90
   adelante :lado
   derecha 90
   adelante :lado
fin
```

Cuando escriban en el Centro de Mando el nombre del procedimiento, seguido del parámetro correspondiente al tamaño del lado (Ejemplo: *cuadrado 100*) debe dibujarse en la pantalla un cuadrado.

Scratch
```
al recibir cuadrado
borrar
bajar lápiz
mover lado pasos
girar 90 grados
mover lado pasos
girar 90 grados
mover lado pasos
girar 90 grados
mover lado pasos
```

Con el fin de poder ejemplificar más adelante las dos formas de ejecutar un procedimiento, se debe escribir este otro procedimiento:

MicroMundos
```
para dibujaCuadrado
   cuadrado 120
fin
```

El procedimiento *cuadrado* ilustra el modelo general para escribir procedimientos con parámetros. Este tipo de procedimientos son muy útiles cuando varias

instrucciones se deben ejecutar en un programa varias veces de manera idéntica, cuya única diferencia sea el valor inicial que tomen algunas variables. En estos casos, las instrucciones se “empaquetan” en un procedimiento y los distintos valores iniciales de las variables se asignan por medio de parámetros.

El nombre del parámetro se debe escribir en la línea del título, después del nombre del procedimiento, ajustándose a las indicaciones establecidas en la Unidad 2 de esta guía para nombrar identificadores. No olvidar que al parámetro se debe anteponer el signo de dos puntos (:) tal como se puede apreciar en el ejemplo 3-4 (*:lado*).

Luego de definir el nombre de un parámetro en la línea de título de un procedimiento (*para cuadrado :lado*), este parámetro se puede utilizar en cualquiera de las instrucciones (ejemplo: *adelante :lado*).

En otras palabras, la línea de título del procedimiento *cuadrado* le dice a Logo que solo hay un parámetro, llamado lado. En el cuerpo del procedimiento se utiliza el Mando *adelante* acompañado del parámetro *:lado*. Esta instrucción dibuja una línea cuya medida es igual al valor que contiene *:lado*.

Cuando en el título de un procedimiento se definen uno o varios parámetros, estos no tienen valor. Solo cuando se ejecuta el procedimiento es que se conoce el valor de cada parámetro. Por esta razón, todo parámetro debe tener un nombre. Por ejemplo, al cambiar la línea de título del procedimiento *cuadrado* por la siguiente (con dos parámetros):

para cuadrado :lado :ángulo

el valor del ángulo se tratará como un parámetro cuando se invoque el procedimiento.

En resumen, un procedimiento, internamente, es un algoritmo que resuelve en forma parcial un problema. Desde el punto de vista externo es una orden o instrucción única que puede formar parte de otro procedimiento o algoritmo (Cajaraville, 1989).

Por otra parte, los estudiantes deben aprender a distinguir entre escribir un procedimiento y ejecutarlo. La primera actividad es el resultado de las etapas de análisis del problema, diseño del algoritmo y traducción de este a un lenguaje de programación.

La segunda actividad (ejecutar un procedimiento) hay dos formas de realizarla. La manera más simple consiste en escribir el nombre del procedimiento en el Centro de Mando (ver ilustración 3-2). Escribir *cuadrado 100* en el Centro de Mando ejecuta el procedimiento llamado *cuadrado*, asigna el valor 100 al parámetro *:lado* y dibuja en la pantalla un cuadrado. Este procedimiento requiere que el usuario especifique la medida de los lados. Para dibujar un cuadrado cuyos lados midan 60 unidades se debe escribir *cuadrado 60* en el Centro de Mando. Para dibujar un cuadrado más grande se debe aumentar el valor del parámetro; por ejemplo:

cuadrado 150.

Otra forma de ejecutar un procedimiento es invocándolo desde otro procedimiento. Cuando se ejecuta el procedimiento *dibujaCuadrado* (sin parámetros), indirectamente se ejecuta el procedimiento *cuadrado* con 120 como parámetro. Este procedimiento dibuja exactamente el mismo cuadrado cada vez que se ejecuta (ver ejemplo 3-4).

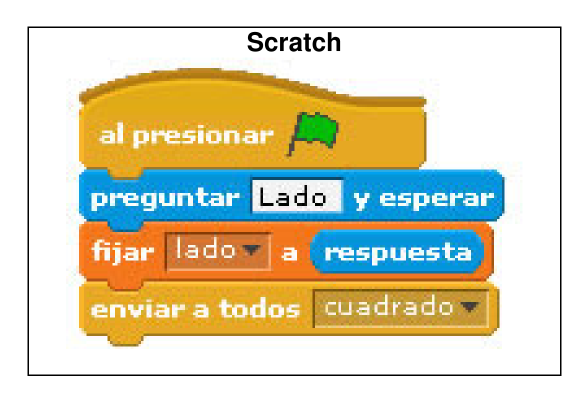
Adicionalmente, poder invocar un procedimiento desde otro, simplifica el dibujo de figuras simétricas. Se dice que una figura es simétrica cuando al doblarla por la mitad sus dos partes coinciden (Casasbuenas & Cifuentes, 1998b). El ejemplo 3-5 en la sección “Estructura Secuencial” ilustra muy bien la construcción de figuras simétricas mediante la utilización de procedimientos que invocan a otros procedimientos.

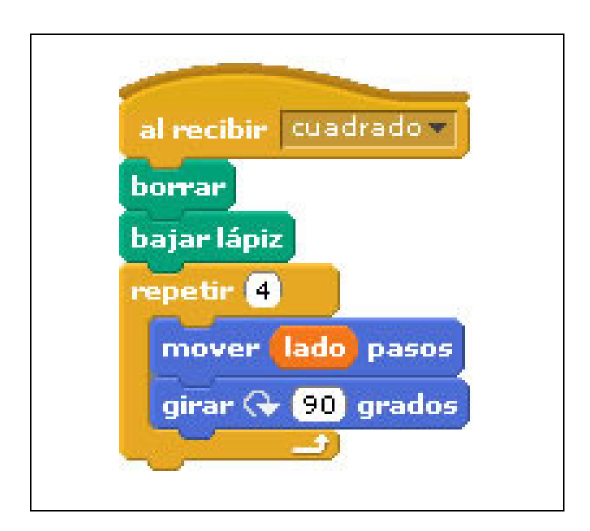
Cada procedimiento tiene su propia biblioteca de nombres, esta es la razón por la cual un procedimiento puede transferirle parámetros a otros, manteniendo los valores de sus propios parámetros. Como los nombres de los parámetros son privados, diferentes procedimientos pueden usar el mismo nombre para un parámetro sin que haya confusión acerca de su valor. Cuando Logo ejecuta un procedimiento, éste establece una biblioteca privada que contiene los nombres de los parámetros que hay en la definición del procedimiento, asociados con los valores dados al momento de llamar al procedimiento. Al ejecutarse una instrucción que contiene el nombre del parámetro, el procedimiento busca en su biblioteca privada el valor de ese nombre. Este es el motivo por el cual el mismo nombre de parámetro puede estar en dos bibliotecas distintas y tener información diferente.

Los parámetros son como "variables locales" del procedimiento en el cual fueron definidos. La importancia de los nombres de entrada privados es que ofrecen la posibilidad de ejecutar un procedimiento sin tener que preocuparse por detalles de su definición.

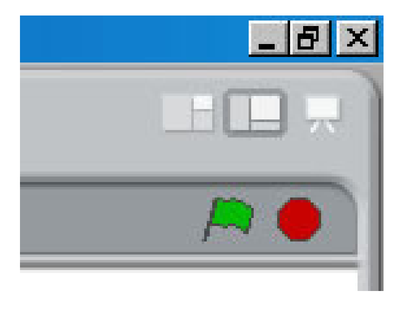
Por su parte, en Scratch no hay procedimientos con parámetros como en MicroMundos; sin embargo, se pueden simular con las instrucciones de Control “Al enviar a todos” y “Al recibir”.

Scratch





Para ejecutar los procedimientos en Scratch basta con hacer clic en la bandera verde ubicada en la esquina superior derecha de la pantalla:



En la siguiente sección se presentan las estructuras de control básicas que los estudiantes deben dominar para poder diseñar algoritmos y traducirlos a un lenguaje de programación.

Una estructura se define como un esquema con cierta distribución y orden que permite representar una idea de forma simplificada y que bajo ciertas condiciones es constante (Trejos, 1999). La estructura de control secuencial es la más sencilla. También se la conoce como estructura lineal. Se compone de instrucciones que deben ejecutarse en forma consecutiva, una tras otra, siguiendo una línea de flujo. Solamente los problemas muy sencillos pueden resolverse haciendo uso únicamente de esta estructura. Normalmente, la estructura secuencial hace parte de soluciones a problemas complejos en las que se la utiliza mezclada con estructuras iterativas (repetir varias veces un conjunto de instrucciones) y condicionales (tomar decisiones).



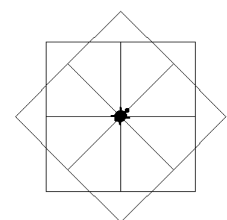
Ilustración 3-6: Modelo de estructura secuencial.

Una estructura de control secuencial puede contener cualquiera de las siguientes instrucciones:

- declaración variables
- asignación de valores
- entrada de datos
- procesamiento de datos (operaciones)
- reporte de resultados

EJEMPLO 3-5

Escribir un procedimiento para dibujar en la pantalla una figura simétrica igual a la siguiente:

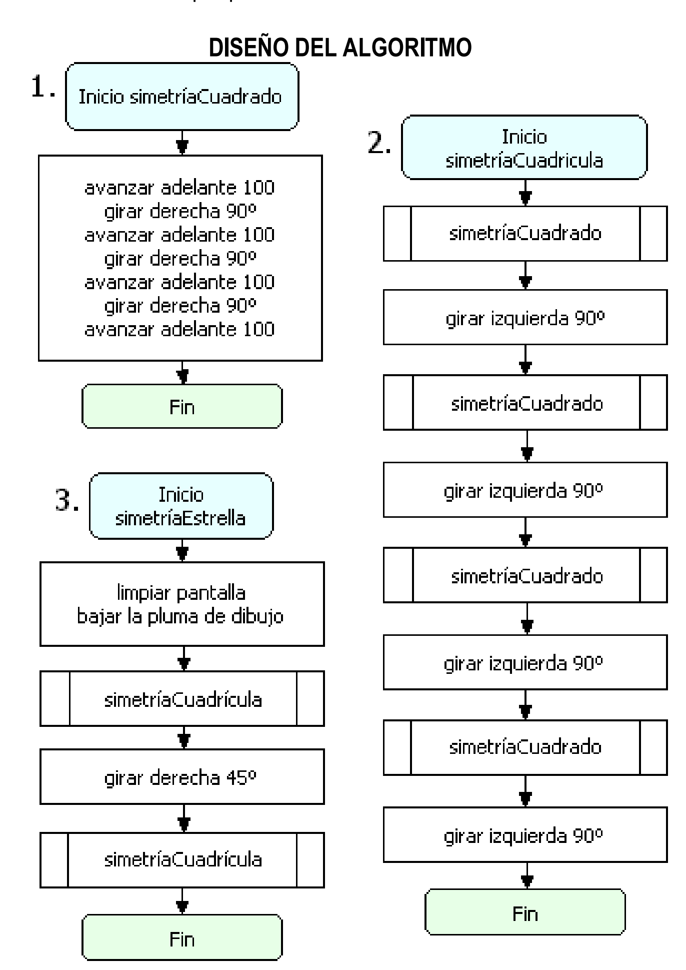


R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Ya está claramente planteado.
Resultados esperados: El dibujo dado.
Datos disponibles: La figura geométrica suministrada. El análisis de la figura permite establecer que está construida con varios cuadrados de igual tamaño.
Restricciones: La figura resultante debe ser igual en su forma a la muestra. Las dimensiones pueden variar.
Procesos necesarios: la figura se puede realizar mediante tres procedimientos. El primero, que puede llamarse *simetríaCuadrado*, dibuja un cuadrado con los comandos *adelante* y *derecha*. El segundo procedimiento (*simetríaCuadricula*), dibuja cuatro cuadrados utilizando el procedimiento *simetríaCuadrado* y el comando *izquierda*. El tercero (*simetríaEstrella*), dibuja dos veces la figura que se forma con el procedimiento *simetríaCuadrado*; se debe girar la tortuga 45 grados a la derecha luego de ejecutar *simetríaCuadrado* por primera vez.

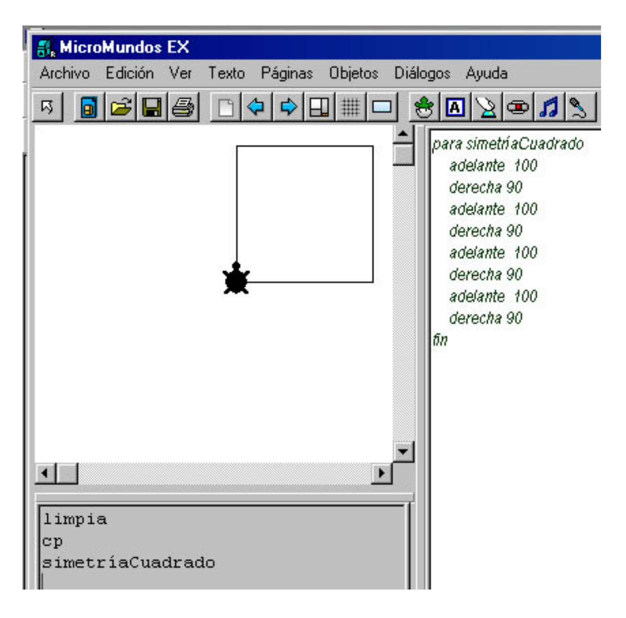
DISEÑO DEL ALGORITMO



TRADUCIÓN DEL ALGORITMO

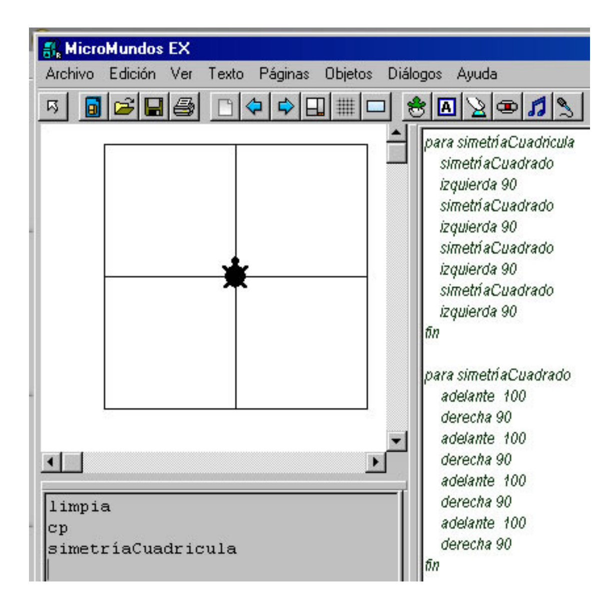
La solución de este problema tiene tres procedimientos:
(1) simetríaCuadrado, (2) simetríaCuadricula y (3) simetríaEstrella.

```
para simetríaCuadrado
   adelante  100
   derecha 90
   adelante  100
   derecha 90
   adelante  100
   derecha 90
   adelante  100
   derecha 90
fin
```



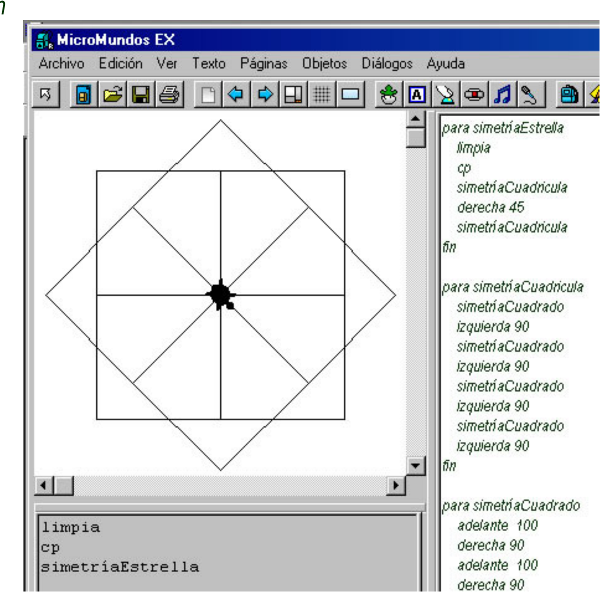
El procedimiento *simetríaCuadrado* (1) está compuesto solamente por una estructura secuencial que realiza el dibujo de un cuadrado cuyos lados miden 100.

para simetríaCuadricula
simetríaCuadrado
izquierda 90
simetríaCuadrado
izquierda 90
simetríaCuadrado
izquierda 90
simetríaCuadrado
izquierda 90
fin



El procedimiento *simetríaCuadricula* (2) está compuesto por una estructura secuencial que dibuja cuatro cuadrados con lados adyacentes entre sí. Esta es una figura simétrica ya que si se dobla por la mitad, ambas mitades coinciden. Para construir esta cuadricula no fue necesario escribir un procedimiento con el código para dibujar cuadro cuadrados; en realidad, el procedimiento *simetríaCuadricula* no realiza ningún dibujo, lo que hace es llamar cuatro veces el procedimiento *simetríaCuadrado* que es el que realmente dibuja un cuadrado cada vez que se invoca. Para que los cuadrados sean adyacentes, se necesita girar la tortuga 90 grados a la izquierda después de cada llamada al procedimiento *simetríaCuadrado*.

para simetríaEstrella
limpia
cp
simetríaCuadricula
derecha 45
simetríaCuadricula
fin



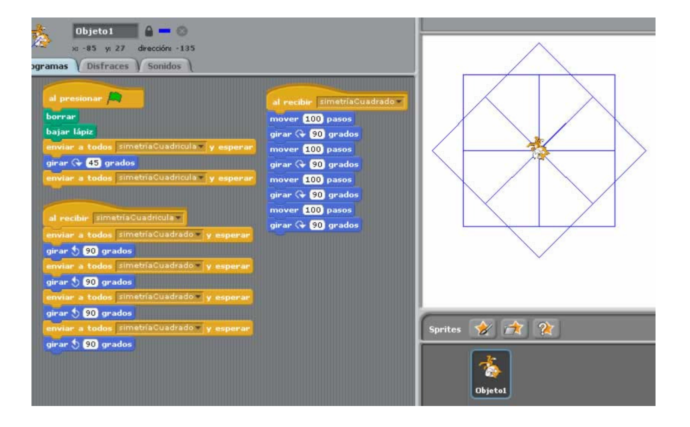
El procedimiento *simetríaEstrella* (3) también está compuesto únicamente por una estructura secuencial que permite dibujar una estrella perfectamente simétrica. Este procedimiento llama dos veces al procedimiento *simetríaCuadricula* que a su vez, cada que es invocado, llama cuatro veces al procedimiento *simetríaCuadrado*. Para lograr el efecto de estrella, luego de llamar la primera vez al procedimiento *simetríaCuadricula*, se gira la tortuga 45 grados, antes de llamar el mismo procedimiento por segunda vez.

Las siguientes son las instrucciones equivalentes en Scratch para elaborar la misma estrella:

```
al presionar [bandera]
borrar
bajar lápiz
enviar a todos simetríaCuadricula y esperar
girar ↻ 45 grados
enviar a todos simetríaCuadricula y esperar
```

```
al recibir [simetríaCuadricula]
enviar a todos [simetríaCuadrado] y esperar
girar ↺ (90) grados
enviar a todos [simetríaCuadrado] y esperar
girar ↺ (90) grados
enviar a todos [simetríaCuadrado] y esperar
girar ↺ (90) grados
enviar a todos [simetríaCuadrado] y esperar
girar ↺ (90) grados
```

```
al recibir [simetríaCuadrado]
mover (100) pasos
girar ↻ (90) grados
mover (100) pasos
girar ↻ (90) grados
mover (100) pasos
girar ↻ (90) grados
mover (100) pasos
girar ↻ (90) grados
```



http://scratch.mit.edu/projects/jualop/42800

Muchos estudiantes logran construir la figura del procedimiento *simetríaEstrella* utilizando gran cantidad de comandos que se repiten sin estructura alguna (mediante experimentación). Es muy importante que ellos reflexionen sobre las ventajas que ofrecen los procedimientos cuando se los utiliza a manera de objetos que cumplen con una función determinada (dibujar un cuadrado, calcular un área, etc). Una tarea que debe realizarse varias veces es candidata ideal para tratarla como un procedimiento. Con la utilización de parámetros se pueden cambiar algunos valores cada vez que se ejecute esa tarea. De esta manera, si necesitamos dibujar varios cuadrados de diferentes tamaños, lo más adecuado será construir un procedimiento con el valor de Lado como parámetro y ejecutarlo varias veces asignando a este el valor del Lado del cuadro a dibujar, cada vez que se ejecute (ver el ejemplo 3-4).

Este ejemplo ilustra la construcción de figuras simétricas mediante la utilización de procedimientos invocados desde otros procedimientos. El concepto de simetría es muy importante tanto en disciplinas como matemáticas y arte como en la cultura general del estudiante. La programación de computadores puede apoyar muy efectivamente el afianzamiento en el niño del concepto que este tiene de simetría, alcanzado en forma intuitiva a través del espejo. “Los espejos son para los niños su primera experiencia y permiten examinar muchos aspectos de las simetrías. Se puede uno preguntar acerca de la inversión mutua entre derecha e izquierda, acerca de las distancias entre el objeto y su imagen en el espejo, lo que ocurre cuando se mueve el objeto o se mueve el espejo, o lo que ocurre cuando estos giran” Fletchet, T. J. citado por Cajaraville (1989).

Por otra parte, la construcción de figuras como estas requiere un dominio espacial del estudiantes ya que es mucho más difícil reproducir una acción correctamente en el pensamiento que llevarla a cabo en el nivel de la conducta. Por ejemplo, es más sencillo moverse de un lugar a otro en un espacio físico o dar vueltas en torno a objetos que representar mentalmente esos movimientos con precisión o representarlos en un plano e invertir mentalmente las posiciones de los objetos haciendo girar el plano (Piaget,1993).

Por último, en este ejemplo se utilizan más comandos de los que realmente se requieren. Como el objetivo no es presentar códigos optimizados sino más bien que el estudiante se familiarice con los comandos disponibles, más adelante, cuando se vea la estructura repetitiva, el docente puede repetir este ejemplo y utilizar el comando “repetir”. Posteriormente, promover la reflexión de los estudiantes sobre la optimización del código y las múltiples maneras que hay en programación para realizar la misma tarea.

EJEMPLO 3-6

Escribir un procedimiento para calcular el área de cualquier triángulo rectángulo. En él se debe pedir al usuario que ingrese los valores de la Altura y la Base del triángulo.

R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Ya está claramente planteado.

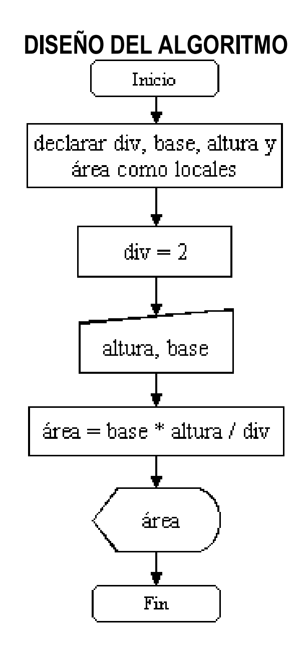
Resultados esperados: Un procedimiento que permita calcular el área de cualquier triángulo rectángulo.

Datos disponibles: Base y Altura del triángulo (se deben solicitar al usuario). El estudiante debe preguntarse si sus conocimientos actuales de matemáticas le permiten resolver este problema; de no ser así, debe plantear una estrategia para obtener los conocimientos requeridos.

Restricciones: Los valores de base y altura son variables y se deben solicitar al usuario.

Procesos necesarios: definir variables; asignar el valor 2 a la constante div; solicitar al usuario el valor de la altura del triángulo; solicitar al usuario el valor de la base; aplicar la fórmula de área; mostrar el resultado.

DISEÑO DEL ALGORITMO



TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

para triánguloRectángulo
local "div ; ***declarar variables y constantes***
local "base
local "altura
local "área
da "div 2
pregunta [Ingrese la Altura del Triángulo] ; ***ingresar altura y base***
da "altura respuesta
pregunta [Ingrese la Base del Triángulo]
da "base respuesta
*da "área :base * :altura / :div* ; ***realizar cálculos***
anuncia frase [El Área del triángulo es:] :área ; ***reportar el resultado***
fin

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

al presionar
fijar div a 2
preguntar Ingrese la altura del triángulo y esperar
fijar altura a 0
preguntar Ingrese la base del triángulo y esperar
fijar base a 0
fijar área a base * altura / div
decir unir El área del triángulo es... área por 2 segundos

En este ejemplo, el procedimiento *triánguloRectángulo* también está compuesto únicamente por una estructura secuencial de instrucciones. En ella se utilizan las primitivas "*pregunta*" y "*respuesta*" para permitir que el usuario del programa suministre al programa los datos "altura" y "base" del triángulo. De esta forma, se logra un procedimiento generalizado para calcular el área de CUALQUIER triángulo rectángulo. En otras palabras, cada vez que se ejecute el procedimiento *triánguloRectángulo*, este le preguntará al usuario cuál es la altura y la base del triángulo del cual desea calcular su área.

Tanto en la utilización de la estructura secuencial, como en las dos que veremos más adelante, es muy importante que los estudiantes reflexionen y determinen el orden de ejecución de las instrucciones (posición) ya que la conmutatividad no es una propiedad aplicable a los algoritmos. El lenguaje algorítmico, al igual que el lenguaje formal de las matemáticas, tiene carácter gráfico y posicional; busca la precisión, el rigor, la abreviación y la universalidad; y, su finalidad fundamental consiste en obtener resultados internamente consistentes (Onrubia & Rochera & Barbarà, 2001).

La construcción de estructuras algorítmicas, entendidas estas como secuencias de instrucciones y operaciones, con el fin de lograr un resultado concreto, ayuda a afianzar en los estudiantes el conocimiento procedimental matemático. Este conocimiento se caracteriza por la acción (saber hacer) frente al conocimiento declarativo que se basa en la enunciación (saber decir). Saber explicar (enunciar) un teorema no garantiza que este se sepa aplicar (actuar) correctamente en la solución de una situación problemática determinada (Onrubia & Rochera & Barbarà, 2001).

Toda secuencia de acciones tiene una estructura que debe planearse (consciente o inconscientemente) antes de ejecutarla. Cuando la acción se realiza de manera automática, quien actúa no es consciente de la estructura y por tanto no puede ver las correlaciones en su actuación y con el entorno de dicha acción; las acciones se convierten en operaciones cuando quien las realiza es consciente de las relaciones inherentes. Pero las acciones prácticas requieren tanta atención que puede ser difícil realizarlas dándose cuenta al mismo tiempo de las correlaciones inherentes a ellas. Por esto, son fundamentales los sistemas de signos a los cuales se traducen las acciones; con los signos se pueden expresar las relaciones que existen dentro de las acciones y entre sus objetos, y se puede proceder con los signos del mismo modo que con los objetos reales (Aebli, 2001).

Según Saussure (1916), citado por Aebli (2001), hay tres grandes grupos dentro de los signos: los símbolos, los signos propiamente dichos y las señales. Un signo, a diferencia de un símbolo, no se parece a su significado; es elegido arbitrariamente y para conocer su significado hay que aprenderlo y fijarlo en la memoria: palabras de lenguajes naturales, cifras, signos algebraicos, etc. Los significados se pueden codificar básicamente de cuatro formas: mediante la palabra hablada, la palabra escrita, el signo gráfico, y la variable. Así, el número 2 se puede representar mediante el fonema "dos", la palabra "dos", el signo "2" o ".." y la variable "a".

De lo anterior se puede deducir que elaborar programas de computador para resolver problemas matemáticos ofrece al estudiante la oportunidad de fijar la atención en la estructura de las operaciones que realiza, mediante su traducción a un sistema de signos que el computador pueda entender. Dirigir la atención a la estructura tiene como consecuencia que operaciones clásicas, como la suma con números naturales, se hagan cada vez más móviles y puedan constituir sistemas cada vez más complejos (Aebli, 2001).

EJEMPLO 3-7

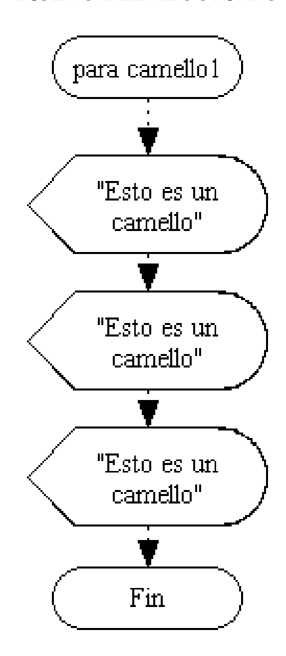
Escribir un procedimiento que muestre 3 veces en pantalla la frase "Esto es un camello".

R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Ya se encuentra claramente formulado.
Resultados esperados: Que aparezca tres veces en pantalla la frase "Esto es un camello".
Datos disponibles: La frase dada.
Restricciones: Ninguna.
Procesos necesarios: Ninguno.

DISEÑO DEL ALGORITMO



TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

```
para camello1
  muestra [Esto es un camello]
  muestra [Esto es un camello]
  muestra [Esto es un camello]
fin
```

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

```
al presionar 🏁
decir Esto es un camello por 2 segundos
decir Esto es un camello por 2 segundos
decir Esto es un camello por 2 segundos
```

Este es un problema muy sencillo de resolver mediante la utilización de una estructura secuencial. Pedir a los estudiantes que analicen qué tan eficiente sería utilizar la misma estructura si el enunciado del problema fuera: Escribir un procedimiento que muestre 60 veces en pantalla la frase "Esto es un camello". ¿Qué habría que hacer si se cambiara 3 veces por 60?

ACTIVIDADES

1. Diseñar un algoritmo que pida al usuario dos números y calcule la suma, la resta, la multiplicación y la división del primero por el segundo. Traducir el algoritmo al lenguaje Logo y probarlo.

2. Diseñar un algoritmo para calcular cuántos litros caben en un tanque. Los datos de entrada (profundidad, largo y ancho) deben estar dados en metros. Se debe tener presente que 1 litro equivale a 1 dm^3. Traducir el algoritmo al lenguaje Logo y probarlo.

ESTRUCTURA ITERATIVA (REPETICIÓN)

La estructura iterativa o de repetición permite ejecutar una o varias instrucciones, un número determinado de veces o, indefinidamente, mientras se cumpla una condición. Esta estructura ayuda a simplificar los algoritmos, ahorrando tiempo valioso a quien resuelve problemas con ayuda del computador.

En programación existen al menos dos tipos de estructuras repetitivas, las cuales a su vez tienen variantes en los diferentes lenguajes de programación. La característica común es que ambos tipos permiten ejecutar una o varias instrucciones:

- un número determinado de veces.
- mientras se cumpla una condición.

Debido a que esta guía está diseñada para educación básica, solo se cubre aquí el primer tipo de estructura repetitiva: Ejecutar una o varias instrucciones un número determinado de veces.

La cantidad de horas disponibles para informática es otro de los factores que impiden desarrollar un programa curricular con todas las variantes que puede ofrecer la estructura repetitiva. Por lo general, la mayoría de instituciones educativas destina una o dos horas semanales a la clase de informática, tiempo muy limitado para cubrir todas las variantes que ofrecen los lenguajes de programación. Ante esta situación, se requiere concentrar esfuerzos en lograr la comprensión de conceptos fundamentales de los tres tipos básicos de estructura (secuencial, repetitiva y condicional). Esto solo se consigue resolviendo una buena cantidad de problemas cuya solución requiera de estas estructuras en forma combinada.

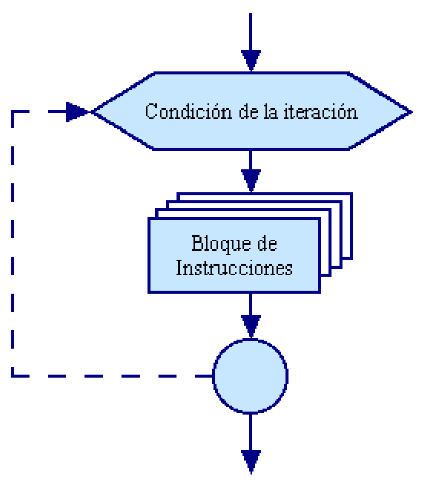


Ilustración 3-7: Modelo de estructura iterativa.

En MicroMundos, la estructura repetitiva se implementa con los Mandos *repite* y *cumpleveces*.

La sintaxis de repite en MMP es:

repite número [lista-de-instrucciones]

ejemplo: *repite 60 [muestra "hola]*
donde *número* (*60*) indica las veces que se ejecutará la *lista-de-instrucciones* (*[muestra "hola]*).

La sintaxis de *cumpleveces* en MMP es:

cumpleveces [serie] [lista-de-instrucciones]

ejemplo: *cumpleveces [i 8][muestra :i]*
donde *serie* (*[i 8]*) indica el nombre del parámetro (*i*) y el número de veces (*8*) que se ejecutará la *lista-de-instrucciones* (*[muestra :i]*).

Por su parte, la estructura repetitiva se implementa en Scratch con los Mandos repetir (n veces); repetir hasta que <una condición sea verdadera>; por siempre; por siempre si <una condición es verdadera>:



EJEMPLO 3-8
Escribir un procedimiento que muestre 85 veces en pantalla la frase "Esto es un camello".
R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Ya se encuentra claramente formulado.
Resultados esperados: Que aparezca 85 veces en pantalla la frase "Esto es un camello".
Datos disponibles: La frase dada.
Restricciones: Ninguna.
Procesos necesarios: Mostrar la frase mencionada 85 veces.

DISEÑO DEL ALGORITMO

Inicio camello2a
repite 1, 85
"Esto es un camello"
Fin

TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

Con el uso del comando repite:

```
para camello2a
  repite 85
      [
       muestra [Esto es un camello]
      ]
fin
```

Ahora con el uso del comando cumpleveces:

```
para camello2b
  cumpleveces [i 85]
    [
     muestra [Esto es un camello]
    ]
fin
```

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

al presionar
repetir 85
decir Esto es un camello por 2 segundos

Un problema similar fue resuelto en el Ejemplo 3-7 de la estructura secuencial. En ese ejemplo se escribió el procedimiento *camello1* que mostraba tres veces en pantalla la frase "Esto es un camello". Tal como debieron advertirlo los estudiantes, resolver este nuevo enunciado agregando instrucciones al procedimiento *camello1* no es práctico. Por eso, en este ejemplo se diseñó un algoritmo muy sencillo mediante la utilización de una estructura iterativa que repite la frase 85 veces. El número de veces que se repite la frase no tiene incidencia en la estructura del algoritmo, sea este 85 ó 1385. Es muy importante que los estudiantes tengan muy claro la diferencia entre los procedimientos *camello1* y *camello2a*.

En el Ejemplo 3-4 se solicitó a los estudiantes dibujar un cuadrado cuyos lados fueran variables. Es muy probable que en una primera aproximación a la solución de este problema ellos elaboren un procedimiento similar al que se planteó en ese ejemplo, utilizando una estructura secuencial:

```
para cuadrado :lado
   limpia
   cp
   adelante :lado
   derecha 90
   adelante :lado
   derecha 90
   adelante :lado
   derecha 90
   adelante :lado
fin
```

Un análisis más elaborado de este problema permitirá a los estudiantes adquirir una conciencia mayor de la organización global de un cuadrado. Determinar los elementos comunes presentes en todos los cuadrados permite identificar qué permanece estático (cuatro segmentos de recta iguales y cuatro ángulos iguales) y qué es lo que cambia (longitud de los segmentos). Aquello que cambia entre un cuadrado y otro se puede tratar como un parámetro. Además, este análisis permite descubrir que es posible utilizar otro tipo de estructura de control para elaborar el mismo dibujo:

TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

```
para cuadrado :lado
   limpia
   cp
   repite 4
     [
     adelante :lado
     derecha 90
     ]
fin
```

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

al presionar
borrar
bajar lápiz
preguntar Tamaño del lado del cuadrado? y esperar
repetir 4
mover respuesta pasos
girar 90 grados

Esta situación evidencia cómo en las actividades de programación el estudiante debe utilizar conocimientos adquiridos con anterioridad, cómo la etapa de análisis favorece y alienta el rigor y la disciplina en el razonamiento y cómo ese análisis puede conducir a nuevos descubrimientos que deriven en reorganizaciones del pensamiento, reestructuraciones de esquemas, etc. (Dufoyer, 1991). Además, algunos

psicólogos han llegado a sugerir que la programación alienta el estudio de las matemáticas, facilita la comprensión de conceptos de esta disciplina, admite explorar activamente campos de conocimiento, permite desarrollar habilidades y ofrece un lenguaje que permite describir la forma personal de resolver problemas.

EJEMPLO 3-9

Calcular el valor de la sumatoria: 1 + 2 + 3 + 4 +5 + … + 100.

R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Ya se encuentra claramente formulado.

Resultados esperados: El resultado de la suma de los números entre 1 y 100.

Datos disponibles: El rango de números dado.

Restricciones: Ninguna.

Procesos necesarios: guardar el número 0 en una variable e incrementarla en 1 cada vez que se ejecute el ciclo repetitivo. Guardar 0 en otra variable e ir acumulando en ella su propio valor más el valor de la primera variable.

DISEÑO DEL ALGORITMO

Inicio
Suma100

contador = 0
sumatoria = 0

repite 1, 100

contador = contador + 1

sumatoria = sumatoria + contador

contador
sumatoria

[El valor de la suma 1 + 2 + 3 + . . . + 100 es]
sumatoria

Fin

Este algoritmo utiliza una operación muy útil en programación:

sumatoria = sumatoria + contador

Consiste en almacenar en una variable *sumatoria* el valor de ella misma (sumatoria) más otro valor variable (*contador*). Es muy utilizada para acumular valores.

TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

Con el Mando repite:

```
para suma100a
  bnombres
  da "contador 0
  da "sumatoria 0
  repite 100
    [
     da "contador :contador + 1
    da "sumatoria :sumatoria + :contador
    muestra nombres
   ]
  muestra frase
    [El valor de la suma 1 + 2 + 3 + . . . + 100 es ] :sumatoria
fin
```

Ahora el mismo algoritmo pero con el Mando cumpleveces de MicroMundos:

```
para suma100b
  bnombres
  da "sumatoria 0
  cumpleveces [contador 100 ]
      [
       da "sumatoria :sumatoria + :contador + 1
      muestra nombres
      ]
  muestra frase
    [El valor de la suma 1 + 2 + 3 + . . . + 100 es ] :sumatoria
fin
```

Los procedimientos *suma100a* y *suma100b* son equivalentes, realizan la misma tarea. La primitiva de MicroMundos *cumpleveces* utilizada en el procedimiento *suma100b* tiene una ventaja adicional con respecto a *repite* : incorpora una variable que aumenta en uno su valor cada vez que se ejecuta un ciclo de la estructura iterativa. La variable, que en este caso se llama *contador* inicia en 0 y termina en 99, para un total de 100 ciclos. Por este motivo se necesita sumarle uno a *contador* para que tome valores entre 1 y 100.

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

```
al presionar [bandera]
fijar contador a 0
fijar sumatoria a 0
repetir 100
  fijar contador a (contador + 1)
  fijar sumatoria a (sumatoria + contador)
  decir sumatoria
  esperar 0.5 segundos
decir unir El valor de la suma 1 + 2 + ... + 100 es = sumatoria por 5 segundos
```

EJEMPLO 3-10

La profesora Ángela Cristina necesita calcular la nota definitiva para cada uno de los 22 alumnos que asisten a su curso de geometría. Ella realizó a todos sus estudiantes, en el primer periodo del año lectivo, dos exámenes y asignó un trabajo de investigación. ¿Cómo puedes ayudarle?

R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Se requiere calcular un promedio de tres notas para cada uno de los 22 alumnos.
Resultados esperados: La nota definitiva de cada uno de los 22 alumnos.
Datos disponibles: El número de alumnos: 22. Las notas de cada alumno las debe digitar la profesora.
Restricciones: Cada una de las tres notas tienen el mismo porcentaje en la nota definitiva. Tres notas por alumno y 22 alumnos.
Procesos necesarios: Para cada uno de los 22 alumnos: Leer las tres notas, sumarlas, calcular el promedio y mostrar el promedio.

DISEÑO DEL ALGORITMO

Inicio
notas
cumpleveces
[estudiante 22]
examenUno, examenDos,
trabajo
sumatoria = examenUno +
examenDos + trabajo
promedio = sumatoria / 3
promedio
Fin

TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

```
para notas
  cumpleveces [estudiante 22]
   [
     pregunta [Ingrese la nota del primer examen ]
     da "examenUno  respuesta
     pregunta [Ingrese la nota del segundo examen ]
     da "examenDos  respuesta
     pregunta [Ingrese la nota del trabajo de investigación ]
     da "trabajo  respuesta
     da "sumatoria :examenUno + :examenDos + :trabajo
     da "promedio :sumatoria / 3
     muestra (frase [El promedio del estudiante ] :estudiante  + 1  [ es
]
       :promedio)
   ]
Fin
```

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

```
al presionar (bandera)
fijar estudiante a 0
repetir 22
  fijar estudiante a (estudiante + 1)
  preguntar Ingrese la nota del primer examen y esperar
  fijar examenUno a respuesta
  preguntar Ingrese la nota del segundo examen y esperar
  fijar examenDos a respuesta
  preguntar Ingrese la nota del trabajo de investigación y esperar
  fijar trabajo a respuesta
  fijar sumatoria a (examenUno + examenDos + trabajo)
  fijar promedio a (sumatoria / 3)
  decir (unir El promedio del estudiante (unir estudiante (unir es promedio)))
  esperar 5 segundos
```

El procedimiento *notas* realiza la misma tarea 22 veces: leer tres notas, sumarlas, promediarlas y mostrar el promedio. La manera más eficiente de resolver este problema es mediante una estructura iterativa. Nótese que los valores de las notas que ingresa el usuario no se validan, esto puede ocasionar que alguien digite una nota de, por ejemplo, 960; lo que dará como resultado un promedio fuera del rango permitido. Este aspecto se atiende con la siguiente estructura, la condicional.

Los ejemplos anteriores ilustran la solución de problemas mediante la utilización de estructuras repetitivas. En ellos se puede apreciar claramente la forma como esta estructura simplifica algunas soluciones, en comparación con soluciones cuya estructura es secuencial.

Por otro lado, la estructura repetitiva es muy apropiada para explorar los conceptos de multiplicación (suma cuyos sumandos son iguales) y potenciación (multiplicación de factores iguales).

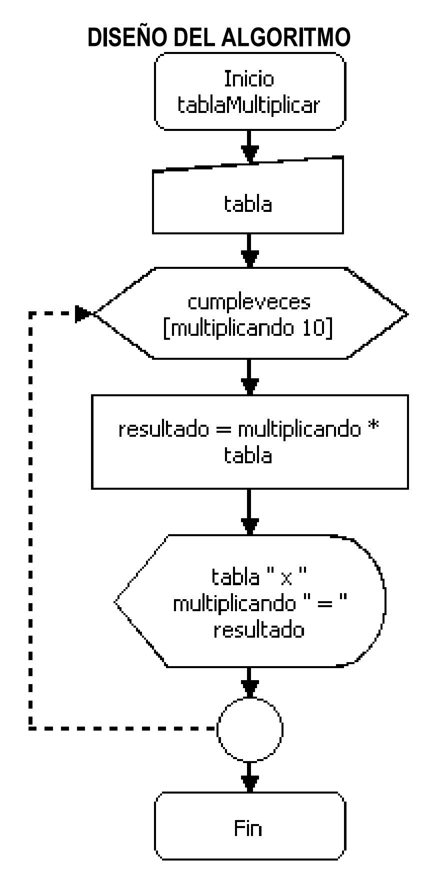
EJEMPLO 3-11

Elaborar un procedimiento para calcular tablas de multiplicar. El usuario debe ingresar qué tabla de multiplicar desea.
R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Ya se encuentra claramente formulado.
Resultados esperados: La tabla de multiplicar que el usuario indique.
Datos disponibles: El número de la tabla (indicada por el usuario).
Restricciones: Ninguna.
Procesos necesarios: pedir al usuario que ingrese la tabla de multiplicar que desea. Guardar ese valor en una variable (tabla). Multiplicar cada uno de los valores entre 0 y 9 por la variable tabla. Mostrar el resultado de cada multiplicación.

DISEÑO DEL ALGORITMO



TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

para tablaMultiplicar
pregunta [¿Qué tabla de multiplicar desea?]
da "tabla respuesta
cumpleveces [multiplicando 10]
[
*da "resultado :multiplicando * :tabla*
muestra (frase :tabla [x] :multiplicando [=] :resultado)
]
fin

Nótese que, al igual que en el ejemplo 3-10, el reportero *frase* devuelve una lista formada por sus entradas (palabras o listas). *frase* puede tomar más de dos entradas cuando se utilizan paréntesis para encerrar el reportero y sus entradas, las cuales pueden incluir texto entre corchetes:
muestra (frase [texto 1] :variable1 [texto2] :variable2 + 1 :variable3 [texto3])

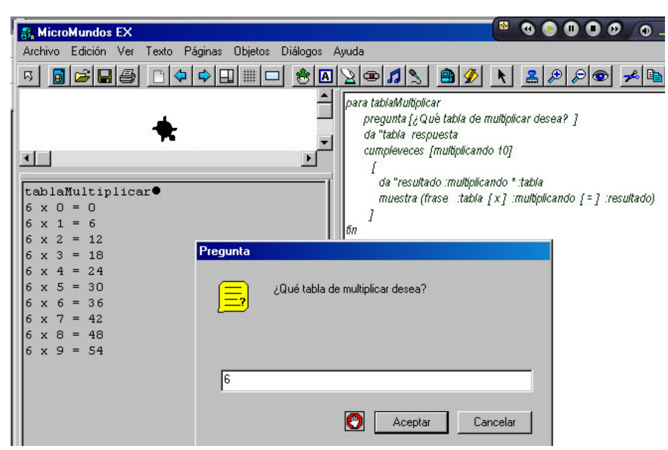


Ilustración 3-8: Ejecución del procedimiento tablaMultiplicar.

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

```
al presionar [bandera]
preguntar [¿Qué tabla de multiplicar desea?] y esperar
fijar [tabla] a (respuesta)
fijar [multiplicando] a [0]
repetir (11)
    fijar [resultado] a ((tabla) * (multiplicando))
    decir (unir (tabla) (unir [x] (unir (multiplicando) (unir [=] (resultado)))))
    esperar (2) segundos
    fijar [multiplicando] a ((multiplicando) + (1))
```

La estructura repetitiva es muy apropiada para reforzar los conceptos de multiplicación y potenciación. La suma 6 + 6 + 6 + 6 + 6 + 6 + 6 puede expresarse como una multiplicación: 6 * 7; de forma similar, la multiplicación 3 * 3 * 3 * 3 puede expresarse como una potencia: 3^4=81. El número 3, que se multiplica varias veces, se conoce como base; 4, la cantidad de veces que se debe multiplicar la base, se conoce como exponente, y 81, el resultado, se conoce como potencia. Se lee: 81 es la potencia de 3 elevado a la 4.

Existe una leyenda muy antigua sobre el origen del juego de ajedrez. La leyenda evidencia claramente la velocidad con que aumenta una progresión geométrica y es un buen punto de partida para empezar a resolver problemas de potenciación con ayuda del computador.

Historia Curiosa

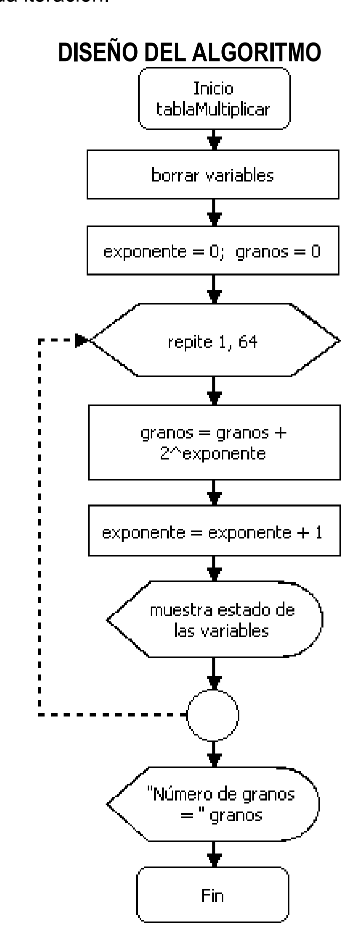
Un día, en la India, un joven bracmán llamado Lahur Sessa pidió una audiencia con el Rey para obsequiarle el juego que había inventado. La curiosidad del rey lo llevó a conceder la cita que pedía el joven Sessa. El rey quedó maravillado y aprendió rápidamente las reglas de aquel juego que consistía de un tablero cuadrado dividido en sesenta y cuatro cuadritos iguales (32 blancos y 32 negros); sobre este tablero se ubicaban dos colecciones de piezas, que se distinguían unas de otras por el color, blancas y negras, repitiendo simétricamente los motivos y subordinadas a reglas que permitían de varios modos su movimiento.
Algún tiempo después, el rey mandó llamar a su presencia al joven bracmán y dirigiéndose a él le dijo:

- Quiero recompensarte, amigo mío, por este maravilloso obsequio, que de tanto me sirvió para aliviar viejas angustias. Pide, pues, lo que desees, para que yo pueda demostrar, una vez más, como soy de agradecido con aquellos que son dignos de una recompensa.
Ante tal ofrecimiento, el joven respondió:
- Voy, pues, a aceptar por el juego que inventé, una recompensa que corresponda a vuestra generosidad; no deseo, sin embargo, ni oro, ni tierras, ni palacios. Deseo mi recompensa en granos de trigo.
-¿Granos de trigo? –exclamó el rey, sin ocultar la sorpresa que le causara semejante propuesta-. ¿Cómo podré pagarte con tan insignificante moneda?
-Nada más simple -aclaró Sessa-. Dadme un grano de trigo por la primera casilla del tablero, dos por la segunda, cuatro por la tercera, ocho por la cuarta, y así sucesivamente hasta la sexagésima cuarta y última casilla del tablero.
No sólo el rey, sino también los visires y venerables bracmanes, se rieron estrepitosamente al oír la extraña solicitud del joven.
Insensato -exclamó el rey-. ¿Dónde aprendiste tan grande indiferencia por la fortuna? La recompensa que me pides es ridícula.
Mando llamar el rey a los algebristas más hábiles de la Corte y les ordenó calculasen la porción de trigo que Sessa pretendía.
Los sabios matemáticos, al cabo de algunas horas de realizar cálculos dispendiosos, volvieron al salón para hacer conocer al rey el resultado completo de sus cálculos.
Preguntóles el rey, interrumpiendo el juego:
-¿Con cuantos granos de trigo podré cumplir, finalmente, con la promesa hecha al joven Sessa?
-Rey magnánimo -declaró el más sabio de los geómetras-: calculamos el número de granos de trigo que constituirá la recompensa elegida por Sessa, y obtuvimos un número cuya magnitud es inconcebible para la imaginación humana (el número en cuestión contiene 20 guarismos y es el siguiente: 18.446.744.073.709. 551. 615. Se obtiene restando 1 a la potencia 64 de 2).
-La cantidad de trigo que debe entregarse a Lahur Sessa -continúo el geómetra- equivale a una montaña que teniendo por base la ciudad de Taligana, fuese 100 veces más alta que el Himalaya. La India entera, sembrados todos sus campos, y destruidas todas sus ciudades, no produciría en un siglo la cantidad de trigo que, por vuestra promesa, debe entregarse al joven Sessa.
¿Cómo describir aquí la sorpresa y el asombro que esas palabras causaron al Rey Ladava y a sus dignos visires? El soberano hindú se veía, por primera vez, en la imposibilidad de cumplir una promesa.
Lahur Sessa -refiere la leyenda de la época-, como buen súbdito, no quiso dejar afligido a su soberano. Después de declarar públicamente que se desdecía del pedido que formulara, se dirigió respetuosamente al monarca y le dijo: los hombres más precavidos, eluden no sólo la apariencia engañosa de los números, sino también la falsa modestia de los ambiciosos.
El rey, olvidando la montaña de trigo que prometiera al joven bracmán, lo nombró su primer ministro.
(Tomado del libro "El hombre que calculaba" escrito por Malba Tahan)

EJEMPLO 3-12
Elaborar un procedimiento para ayudar a los hábiles algebristas de la corte del Rey Ladava con el cálculo del número de granos de trigo que deben entregar a Lahur Sessa como pago por haber inventado el juego de ajedrez.
R/.

ANÁLISIS DEL PROBLEMA
Formular el problema: Es un problema de multiplicaciones de factores iguales que pueden expresarse en forma de potencias; además, para llegar al resultado final se deben acumular los resultados parciales.
Resultados esperados: El número de granos que el Rey Ladava debe entregar a Lahur Sessa.
Datos disponibles: El número de cuadros del tablero de ajedrez (64) y la regla dada por Sessa: "un grano de trigo por la primera casilla del tablero, dos por la segunda, cuatro por la tercera, ocho por la cuarta, y así sucesivamente hasta la sexagésima cuarta y última casilla del tablero".
Restricciones: Aplicar la regla planteada por Sessa.
Procesos necesarios: Un ciclo que se repita 64 veces. En cada iteración se debe acumular en una variable (granos), su propio valor más el resultado de 2 elevado a un exponente que aumenta su valor en uno con cada iteración.

DISEÑO DEL ALGORITMO



TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS
Utilizando el comando repite:

```
para ajedrez
   bnombres
   da "exponente  1
   da "granos 0
   repite 64
      [
        da "granos :granos + potencia 2 :exponente
       da "exponente :exponente + 1
       muestra nombres
      ]
```

muestra frase
[El número de granos que ganó Sessa es] :granos
fin

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

```
al presionar [bandera verde]
fijar exponente a 0
fijar granos a 0
repetir 64
    enviar a todos potenciación y esperar
    cambiar granos por potencia
    cambiar exponente por 1
decir unir El número de granos que ganó Sessa es granos
esperar 5 segundos
```

```
al recibir potenciación
fijar base a 0
fijar potencia a 1
repetir exponente
    fijar base a potencia * 2
    fijar potencia a base
```

Nótese que como Scratch (hasta la versión 1.4) no incluye un operador para calcular potencias, una forma de solucionar esta situación consiste en elaborar un subprocedimiento llamado potenciación para realizar este cálculo a partir de multiplicaciones sucesivas.

¿Obtuvieron los estudiantes el mismo resultado (18446744073709551615)?
¿El computador también tardó varias horas para calcular el resultado final?
¿Qué función cumplen los comandos bnombres y nombres en MicroMundos?
¿Cuál es la diferencia entre crecimiento aritmético y crecimiento geométrico?

ACTIVIDADES
Los estudiantes deben encontrar solución a los siguientes problemas empleando la metodología expuesta en la Unidad 1: Analizar el problema (formulación del problema, resultados esperados, datos disponibles, restricciones y procesos necesarios), diseñar el algoritmo, traducirlo al lenguaje Logo y probar el programa resultante.

1. Elaborar un procedimiento que calcule y muestre las áreas de 100 círculos con radio de 1 a 100 cm.

2. Elaborar un procedimiento que calcule y muestre el cuadrado de los números 1 a 90.

3. Elaborar un procedimiento que le reporte al electricista de un edificio recién construido cuantos bombillos debe comprar. Se sabe que el edificio tiene 8 pisos, 8 apartamento en cada piso y cada apartamento tiene 8 bombillos. En la solución se debe emplear una estructura repetitiva.

4. Elaborar un procedimiento que calcule el área de cualquier cubo.

5. Elaborar un procedimiento que dibuje polígonos regulares de 5, 6, 7, 8 y 9 lados. El usuario debe indicar el número de lados del polígono.

ESTRUCTURA CONDICIONAL

Es fundamental que los estudiantes presten atención especial a las estructuras que utilizan para resolver problemas y las reconozcan para lograr mayor control sobre la solución planteada. De esta manera, la programación de computadores les ayuda a planear conscientemente las secuencias de acciones que resuelven un problema planteado y las estructuras involucradas en una solución dada.

La estructura condicional se utiliza para indicarle al computador que debe evaluar una condición y, a partir del resultado, ejecutar el bloque de instrucciones correspondiente. La forma más común está compuesta por una proposición (condición) que se evalúa y dos bloques de instrucciones que se ejecutan, uno cuando la condición es verdadera (selección simple y doble) y otro cuando ésta es falsa (únicamente en la selección doble). Algunos autores se refieren a este tipo de estructura como estructura de selección, estructura selectiva o estructura de decisión; en esta guía, todas estas denominaciones son consideradas sinónimas.

Para que una proposición (frase declarativa) sea valida, debe poder afirmarse que es verdadera o falsa. En programación, se utilizan operadores relacionales (<, =, >) para establecer la relación que existe entre dos elementos de la proposición. Por ejemplo, "La calificación de Esteban en Historia es mayor que 6.0", es una proposición valida. De una parte tenemos "*La calificación de Esteban en Historia*" (A) y, de la otra, el valor "*6.0*" (B); de A con respecto a B, se afirma que "A es mayor que B", por lo tanto, la relación existente entre A y B es "ser mayor que". Para que el computador entienda esta proposición, debe expresarse así: "*:calificación > 6.0*", donde *:calificación* es la variable que contiene el valor de "la calificación de Esteban en Historia".

OPERADOR	DESCRIPCIÓN	EJEMPLO
=	Igual que	:ánguloUno = 90 :tipo = "SI
<	Menor que	:ánguloUno < 90
>	Mayor que	:ánguloUno > 90

Adicionalmente, las proposiciones pueden ser sencillas o compuestas. Las proposiciones compuestas se forman con dos o más proposiciones sencillas unidas por operadores lógicos (y, o, no). Cuando se unen dos proposiciones por medio del operador lógico "y", significa que ambas proposiciones deben ser verdaderas (conjunción). Cuando se unen dos proposiciones por medio del operador lógico "o", significa que por lo menos una de las dos proposiciones debe ser verdadera (disyunción).

Por su parte, un bloque de instrucciones puede contener una o varias instrucciones que se ejecutan una detrás de otra. La estructura condicional tiene tres variantes:

- selección simple.
- selección doble.
- selección múltiple.

Las estructuras condicionales simple y doble evalúan una proposición (condición) que devuelve como resultado únicamente dos valores posibles y excluyentes: verdadero o falso. En cambio, la estructura condicional de selección múltiple permite que la condición devuelva más de un valor posible y que para cada uno de esos valores se ejecute el bloque de instrucciones correspondiente. Por ejemplo, una situación típica de selección múltiple es cuando la incorporación al ejercito, de un joven al terminar sus estudios de educación media, depende del color de una balota: si saca una balota roja, su incorporación al ejercito es inmediata; si es azul, la incorporación será en julio; y si es blanca, el estudiante no debe prestar servicio militar. En esta situación hay tres valores posibles y cada uno de esos valores implica la ejecución de una instrucción diferente (Jiménez, 2002).

Debido al alcance de esta guía, solo se cubren aquí los dos primeros tipos de estructura condicional: simple y doble.

Selección simple

La estructura condicional de selección simple ejecuta un bloque de instrucciones cuando la proposición (condición) es verdadera; si esta es falsa, no hace nada.

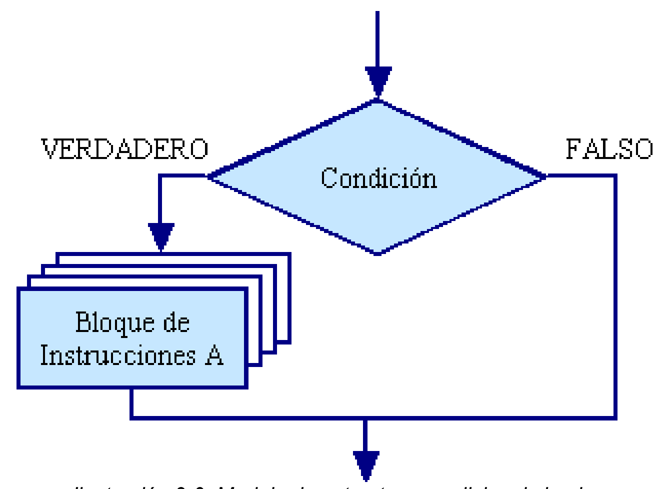


Ilustración 3-9: Modelo de estructura condicional simple.

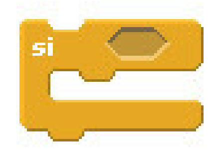
Para la estructura condicional de selección simple, MicroMundos ofrece el comando "*si*". La sintaxis es:

si cierto-o-falso

[
lista-de-instrucciones
]

el comando "*si*" ejecuta la lista-de-instrucciones únicamente si al evaluarse la proposición, esta devuelve cierto (verdadero).

Por su parte, la estructura condicional de selección simple se implementa en Scratch con el bloque "si" (condición):



EJEMPLO 3-13

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

al presionar
preguntar Ingrese el ángulo y esperar
fijar ángiloUno a respuesta
si ángiloUno = 90
decir Es un ángulo recto!! por 2 segundos

TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

```
para selecciónSimple
   pregunta [Ingrese el ángulo]
   da "ánguloUno respuesta
   si :ánguloUno = 90
       [
        da "reportar [ es un ángulo recto]
        muestra frase :ánguloUno :reportar
       ]
fin
```

En este ejemplo, *cierto-o-falso* (*:ánguloUno = 90*) indica la condición que se debe evaluar la cual puede devolver únicamente uno de dos valores posibles: verdadero o falso. En caso de ser verdadera la proposición, se ejecuta la *[lista-de-instrucciones]* indicada entre corchetes; esta puede contener una o varias instrucciones. Cuando es falsa la proposición evaluada, no se ejecutan instrucciones.

Además, se puede observar un recurso gráfico muy importante para dar claridad a las líneas de código de los procedimientos en MicroMundos: (1) dejar líneas en blanco para dividir bloques de código; (2) utilizar sangrías para indicar porciones de código subordinadas a un comando; (3) abrir y cerrar los corchetes que indican bloques de código en una línea a parte, de tal forma que se aprecie muy claramente dónde inicia y dónde termina una *lista-de-instrucciones*.

Selección doble

La estructura condicional de selección doble ejecuta un bloque de instrucciones (A) cuando la proposición (condición) es verdadera y un bloque diferente (B) cuando esta es falsa.

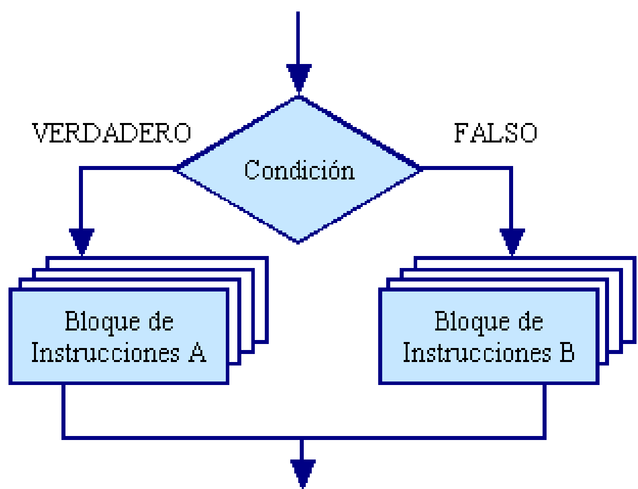


Ilustración 3-10: Modelo de estructura condicional doble.

Para la estructura condicional de selección doble, MicroMundos ofrece el comando "*siotro*". La sintaxis es:

siotro cierto-o-falso
[
lista-de-instrucciones-A
]
[
lista-de-instrucciones-B
]

El comando "*siotro*" ejecuta la *lista-de-instrucciones-A* si al evaluarse la proposición, esta es verdadera. Si la proposición es falsa, se ejecuta la *lista-de-instrucciones-B*. Ambas listas de instrucciones se deben indicar entre corchetes [ ] y pueden estar compuestas por una o más instrucciones.

En Scratch, la estructura condicional de selección doble se implementa con el bloque "si (condición) si no":



EJEMPLO 3-14

```
para selecciónDoble
   pregunta [Ingrese el ángulo]
   da "ánguloUno respuesta
   siotro (:ánguloUno = 90)
      [
       da "reportar [ es un ángulo recto]
```

```
    ]
    [
      da "reportar [ NO es un ángulo recto]
    ]
  muestra frase :ánguloUno :reportar
fin
```

```
al presionar
preguntar [Ingrese el ángulo] y esperar
fijar [ángiloUno] a respuesta
si ángiloUno = 90
  decir [SI es un ángulo recto!!] por 5 segundos
si no
  decir [NO es un ángulo recto!!] por 5 segundos
```

En este ejemplo, *cierto-o-falso* (*:ánguloUno = 90*) indica la proposición que se debe evaluar, la cual solo puede devolver uno de dos valores posibles: verdadero o falso. En caso de que la proposición sea verdadera, se ejecuta la *[lista-de-instrucciones-A]* indicada entre corchetes: (*[da "reportar [es un ángulo recto]]*). Cuando la proposición evaluada es falsa, se ejecuta la *[lista-de-instrucciones-B]* (*[da "reportar [NO es un ángulo recto]]*).

Nótese que en MicroMundos la instrucción
muestra frase :ánguloUno :reportar
se encuentra fuera de los corchetes; por tanto, se ejecutará sin importar si la proposición es verdadera o falsa. Además, ejemplifica muy bien el concepto de variable ya que el valor del ángulo se guarda en la variable denominada *ánguloUno* y el aviso que se debe mostrar acerca de si el ángulo es o no recto, también se guarda en una variable (*reportar*).

Tanto en la estructura de selección simple como en la doble se debe tener en cuenta lo siguiente:

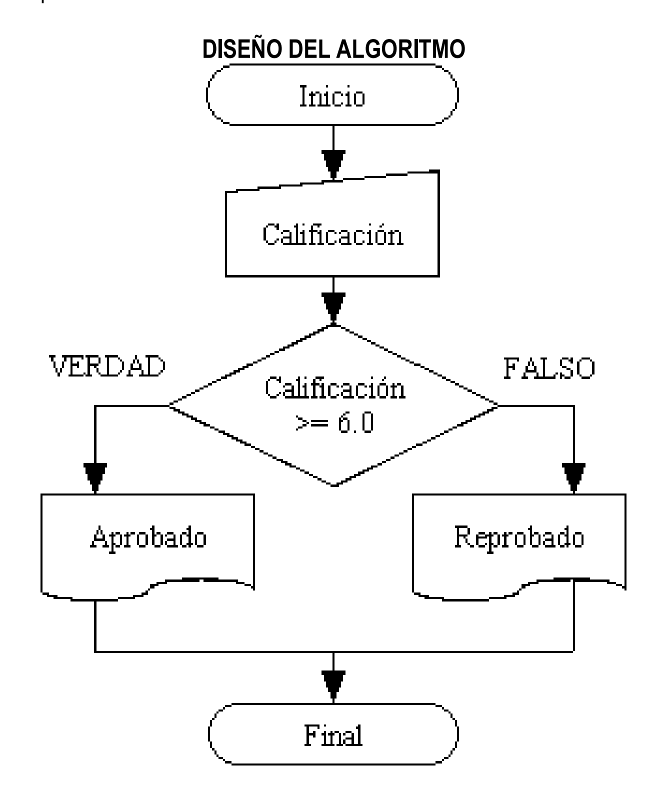
- La proposición debe ser una frase declarativa, la cual se pueda afirmar o negar.
- En MicroMundos, se requiere que en el encabezado vayan las palabras reservadas *si* y *siotro* respectivamente.
- En MicroMundos, cuando la proposición es sencilla (sin operadores lógicos) no es necesario que vaya entre paréntesis; si es compuesta (dos o más proposiciones unidas con operadores lógicos como: o, y, no) tiene que encerrarse con paréntesis. Como en el primer caso no sobran los paréntesis (no genera error), es recomendable utilizarlos siempre. Por ejemplo: *(ánguloUno = 90)* es una proposición sencilla equivalente a *ánguloUno = 90*, pero es mejor utilizar la primera forma.
- En MicroMundos, las listas de instrucciones deben estar agrupadas con corchetes, estos indican dónde empieza y dónde termina la lista que conforma el bloque que se debe ejecutar.

EJEMPLO 3-15
Un estudiante aprueba un examen cuando obtiene una calificación mayor o igual a seis. Elaborar un procedimiento que pida al usuario una calificación, aplique el criterio de aprobación e imprima "Aprobado" o "Reprobado", según sea el caso.
R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Es un problema sencillo de selección doble.
Resultados esperados: Un aviso que reporte si el estudiante "Aprobó" o "Reprobó" el examen.
Datos disponibles: La calificación ingresada por el usuario. Para aprobar, la nota debe ser mayor o igual a 6.0.
Restricciones: Aplicar el criterio de aprobación.
Procesos necesarios: Solicitar al usuario que ingrese la calificación. Evaluar si la calificación es igual o superior a 6.0; en caso de ser verdadero, reportar "Aprobado"; en caso contrario, reportar "Reprobado".

DISEÑO DEL ALGORITMO



TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

```
para aprueba
  local "calificación
  pregunta [Ingrese la Calificación]
  da "calificación respuesta
  siotro o :calificación > 6.0  :calificación = 6.0
      [
       anuncia [Aprobado]
      ]
      [
       anuncia [Reprobado]
     ]
fin
```

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

```
al presionar [bandera]
cambiar el disfraz a [cassy-chillin]
preguntar [Ingrese la calificación] y esperar
fijar [calificación] a (respuesta)
si <<(calificación) > [6.0]> o <(calificación) = [6.0]>>
    cambiar el disfraz a [feliz]
    decir [Aprobado!!!!!!! :)] por (5) segundos
si no
    cambiar el disfraz a [triste]
    decir [Reprobado :(] por (5) segundos
```

En este ejemplo, se puede observar la sintaxis de los operadores lógicos (y, o, no), mediante los cuales se unen proposiciones sencillas para construir proposiciones compuestas. Estos deben ir en seguida del paréntesis que abre la proposición:
siotro o :calificación > 6.0 :calificación = 6.0
La proposición se lee así:
"calificación mayor que 6.0 o calificación igual a 6.0".

Proposiciones

Según Piaget (1993), las operaciones verbales o proposicionales surgen hacia los once o doce años con la capacidad para razonar por hipótesis. Esta capacidad hipotética-deductiva es la que hace posible que los niños entre los once y los catorce años piensen en términos de proposiciones y no únicamente sobre objetos; acepten cualquier tipo de dato como puramente hipotético y razonen correctamente a partir de él; deduzcan las implicaciones de enunciados posibles y así distingan entre lo posible y lo necesario; formulen todas las hipótesis posibles relativas a los factores que entran en juego en una actividad y organicen la información en función de estos factores.

De acuerdo con Piaget y sus seguidores, es en este estado del desarrollo cognitivo en el que se constituye un conjunto de estructuras proposicionales basadas en lo que en teoría de conjuntos se llama un "conjunto de todos los subconjuntos". Esta estructura está conformada por operaciones equivalentes a ciertas estructuras del pensamiento verbal, tales como implicación (p -> q: si..., entonces...; si la hipótesis p es verdadera, entonces la consecuencia q se sigue necesariamente); disyunción (p V q; ó p, ó q, ó los dos); unión (p ^ q); incompatibilidad (p | q).

Una forma efectiva para iniciar a los estudiantes más pequeños en el tema de las proposiciones puede ser la propuesta por Marquínez & Sanz (1988): empezar con cadenas de palabras (sin sentido), avanzar a expresiones (con sentido incompleto), continuar con oraciones (con sentido completo) y finalizar con proposiciones simples y compuestas (calificables como falsas o verdaderas).

EJEMPLO

"La escuela tiene pan francés caído de China" es una CADENA de palabras que carece de sentido.
"Los amigos de lo ajeno" es una EXPRESIÓN que tiene sentido pero no completo.
"Ojalá que mañana no llueva" es una ORACIÓN con sentido completo pero no es calificable.
"Simón Bolívar nació en Santa Marta" es una PROPOSICIÓN que puede calificarse de verdadera o falsa.

ACTIVIDAD

Escribir en el espacio si la propuesta corresponde a una cadena, expresión, oración o proposición:

a. ________________ prohibido fumar en el salón de clase.
b. ________________ el oro es un elemento de la tabla periódica.
c. ________________ calle perfecta para perro azul.
d. ________________ el carro sedán azul.
e. ________________ ¿qué hora es?
f. ________________ el nevado del Ruiz es un volcán.
g. ________________ Simón Bolívar murió en Santa Marta
h. ________________ Cali es una ciudad colombiana.
i. ________________ camisa cuadrada por carro naciente.
j. ________________ Perú y Chile son países Iberoamericanos.
k. ________________ el cuaderno verde de geometría.
l. ________________ está permitido subir las escaleras.
m. ________________ cuatro y diez son números menores que veinte.
n. ________________ si alguien es chileno, entonces es español.
o. ________________ hace mucho frío
p. ________________ en un lugar de la Mancha de cuyo nombre
q. ________________ ojalá no me llame.
r. ________________ apague la luz cuando salga.

Un curso de algoritmos y programación puede contribuir significativamente a desarrollar la capacidad hipotética-deductiva en la que el pensamiento no proceda de lo real a lo teórico, sino que parta de la teoría y establezca o verifique relaciones reales entre cosas. Concretamente, dos tipos de actividades pueden ayudar a lograr este propósito: utilizar estructuras condicionales las cuales están basadas en la operación de implicación (si...., entonces...) y formular enunciados declarativos compuestos (proposiciones simples unidas por los conectores lógicos "y", "ó") que el computador pueda evaluar como verdaderos o falsos. Adicionalmente, estos enunciados promueven el razonamiento por atribución o relación (Felipe es más joven que Ángela) en contraposición al razonamiento por predicados (Felipe es joven).

Precisamente, la estructura condicional utilizada en programación (si... entonces...) ofrece al estudiante oportunidades para desarrollar habilidades con proposiciones y relaciones de orden. Sin embargo hay que tener en cuenta que la construcción "si P entonces

S", que utilizan los lenguajes de programación MicroMundos y Scratch, es procedimental y no declarativa ya que hace énfasis en la acción y no en el concepto semántico de verdad (Iranzo, 2005). Mientras que en lógica se indica que entre P y S hay una relación de dependencia en la que al suceder P, necesariamente se causa S; en programación se indica que cuando P es verdadero, necesariamente se ejecuta un conjunto de instrucciones A y en caso de ser falso no se ejecuta ninguna instrucción (selección simple) o necesariamente se ejecuta un conjunto de instrucciones B (selección doble).

Según Bustamante (2007), "una proposición es una frase declarativa que puede ser afirmada o negada" y para Iranzo (2005) la lógica proposicional "se ocupa de los enunciados declarativos simples como un todo indivisible y que pueden combinarse mediante partículas lógicas denominadas conectores (no, y, o, si... entonces..., etc)". A esta lógica también se le conoce con el nombre de lógica de enunciados o lógica de conectores. De acuerdo con estos dos autores, los siguientes enunciados declarativos se pueden negar o afirmar, por lo tanto pueden considerarse proposiciones:

1. Cali es la capital del Valle del Cauca.
2. El cuatro es un número impar.
3. Seis es menor que doce.
4. El INSA es un colegio regentado por la comunidad de Padres Basilianos.
5. Andrés Pastrana es el presidente de Colombia.
6. Es verano
7. Hace calor

De las proposiciones primera, tercera, cuarta y quinta podemos decir que son verdaderas y de la segunda podemos afirmar que es falsa. Sin embargo, para poder afirmar que la cuarta proposición es verdadera, hay que disponer del conocimiento suficiente sobre este colegio ubicado en el barrio Andrés Sanín en la ciudad de Cali. Esto nos conduce a hacer otra consideración: establecer explícitamente si una proposición es verdadera o falsa puede resultar en algunos casos muy difícil o imposible. Por otra parte, la quinta proposición fue verdadera durante un lapso de tiempo (1998-2002).

En relación a las proposiciones sexta y séptima, su valor de verdad depende del momento en el cual se haga la afirmación. Esto nos lleva a otra forma de clasificar los enunciados declarativos: de acción cuando el sujeto no está determinado (6 y 7); de atribución cuando el sujeto es determinado y se le atribuye una propiedad (1, 2 y 5); y de relación cuando hay dos o más sujetos (3 y 4).

Con respecto a las relaciones de orden podemos decir que consisten en un par de elementos presentes en una proposición relacionados por medio de un atributo gradado. Por ejemplo, "el elemento A es *mayor o igual que* el elemento B" o "seis es *menor que* doce". Proposiciones en las cuales "mayor o igual que" y "menor que" son las relaciones de orden que se establecen entre los elementos A y B, y entre seis y doce, respectivamente.

Hay que tener cuidado con el uso del lenguaje cotidiano en el que dos relaciones pueden ser equivalentes como "igual o superior a" y "mayor o igual que". En cambio, "entre 0 y 10, inclusive" y "entre 0 y 10" no son equivalentes; en la primera relación los valores 0 y 10 hacen que la proposición sea verdadera, en la segunda relación, no.

RELACIÓN	**(MicroMundos) y Scratch**
A es igual a B	(:a = :b)
A es mayor que B	(:a > :b)
A es mayor o igual que B	(o :a > :b :a = :b)
A es como mínimo igual a B	(o :a > :b :a = :b)
A es menor que B	(:a < :b)
A es menor o igual que B	(o :a < :b :a = :b)
A es al menos igual a B	(o :a < :b :a = :b)
A está entre 0 y 10	(y :a > 0 :a < 10)
A está entre 0 y 10, inclusive	(y (o :a > 0 :a = 0) (o :a < 10 :a = 10))

Un aspecto fundamental de la estructura condicional es la reflexión sobre el papel del lenguaje natural en la formulación y uso de relaciones de orden y de proposiciones. Diversos autores que se han ocupado de la lógica y el lenguaje han establecido tres categorías generales para el uso del lenguaje: informativa (suministra información definiendo, declarando, aclarando, describiendo), expresiva (expresa sentimientos, emociones, deseos) y directiva (busca inducir a alguien a que haga u omita algo). Son ejemplos de cada una de estas categorías lo siguiente:
Uso informativo:

- La línea recta es la más corta entre dos punto de un plano.
- Colombia es un país andino
- Los noruegos son altos, delgados y de ojos azules.

Uso expresivo:

- Ojalá haga buen día mañana!
- Qué horror! no podría soportar algo tan doloroso.

Uso directivo:

- Prohibido fumar
- Cierre la puerta
- Se solicita comportarse bien

Para la programación y en especial para la estructura condicional, resulta imprescindible el uso informativo del lenguaje. Este se encarga de comunicar información mediante la formulación y afirmación o negación de proposiciones. El discurso informativo se utiliza para describir el mundo y para razonar sobre él, sin importar si las proposiciones son importantes o no, si son generales o específicas, o si son verdaderas o falsas (Copi & Cohen, 2000).

Los estudiantes deben estar en capacidad de distinguir el discurso informativo en un texto o en el planteamiento verbal de un problema. Pero en ciertos textos o planteamientos resulta difícil identificar de manera inmediata la existencia de proposiciones que se puedan contestar con un "verdadero" o con un "falso" (Solano, 1991). En lenguajes de programación como Logo es muy importante que las proposiciones se puedan expresar directamente, en forma de notación matemática o mediante texto. Para ello, es fundamental que los estudiantes identifiquen los componentes de las proposiciones (enunciados y relación entre ellos) y verifiquen que sean validos. Luego determinen en cada proposición el sujeto (objetos o individuos acerca de los cuales se afirma algo) y el predicado (propiedad que posee el sujeto) y en seguida identifiquen con un nombre (identificador) al que puede variar (sujeto o predicado).

Por ejemplo, en la proposición número 1 "Cali es la capital del Valle del Cauca", el sujeto es "Cali", el predicado es "capital del Valle del Cauca" y la relación es de igualdad "es". Se debe asignar un nombre al predicado (capitalValle) para guardar el valor "Cali". En el caso de la proposición "Seis es menor que doce", el sujeto es "Seis", el predicado es "doce" y la relación es "menor que".

Por otra parte, de las siete proposiciones planteadas, solo la número tres se puede expresar en notación matemática; la otras proposiciones hay que expresarlas como texto, con excepción de la número 2 que no se puede expresar directamente:

1. (:capitalValle = "Cali)
2. "El cuatro es un número impar" no se puede expresar directamente. Hay que elaborar un procedimiento para determinar si un número es par o impar.
3. (6 < 12)
4. (:rectorINSA = "Basiliano)
5. (:presidenteColombia = "|Álvaro Uribe|)
6. (:verano = true)
7. (:haceCalor = false)

TIP

Hay que tener cuidado cuando se copia de un procesador de texto una porción de texto que contenga comillas (") y se pega en el área de procedimientos de MicroMundos [4]. Las comillas (") que generan estos programas no son equivalentes a las comillas de MicroMundos [4] (").

También hay que tener cuidado cuando se quiere comparar un texto conformado por dos o más palabras, este debe encerrarse entre barras (|palabra1 palabra2|).

Otro aspecto a tener en cuenta con las proposiciones y que se debe trabajar con los estudiantes es la riqueza del lenguaje natural (Marquínez & Sanz, 1998). Por ejemplo, el conector lógico "y" (^) se presenta de diversas formas en el lenguaje común utilizado para formular problemas y los estudiantes deben aprender a identificarlo:

- Cali Y Medellín son ciudades ecuatorianas.
- Bogotá, Quito, Lima, Montevideo son ciudades capitales (*Bogotá es ciudad capital Y Quito es ciudad capital Y Lima es ciudad capital Y Montevideo es ciudad capital*)
- Luisa estudia, Cristina también (*Luisa estudia Y Cristina estudia*)
- En Bogotá hace frío, IGUALMENTE en Tunja (*En Bogotá hace frío Y en Tunja hace frío*)
- En Bogotá hace frío, DEL MISMO MODO en Tunja (*En Bogotá hace frío Y en Tunja hace frío*)
- En Bogotá hace frío, MIENTRAS QUE en Cartagena calor (*En Bogotá hace frío Y en Cartagena hace calor*)
- Ángela tiene un automóvil, PERO no sabe manejarlo aún (*Ángela tiene automóvil Y Ángela no sabe manejar automóvil*)
- Luisa no viene, SIN EMBARGO escribe correos electrónicos todos los días (*Luisa no viene Y Luisa escribe correos electrónicos todos los días*)
- Esteban no estudia, NO OBSTANTE quiere hacerlo (*Esteban no estudia Y Esteban quiere estudiar*)
- A PESAR DEl buen tiempo, no vamos a la piscina (*Hace buen tiempo Y no vamos a piscina*)
- PESE A QUE lo sabe, no lo puede decir (*Él lo sabe Y él no lo puede decir*)
- En Cali no hay energía eléctrica, TAMPOCO en Bogotá (*En Cali no hay energía eléctrica Y en Bogotá no hay energía eléctrica*)

Lo mismo ocurre con la determinación de si una proposición está expresada en afirmativo o en negativo:

- Colombia NO es un país europeo.
- El Nilo es un río Incontrolable (*El Nilo es un río que NO se puede controlar*)
- La vida humana en Marte es Imposible (*NO es posible la vida humana en Marte*)
- Luisa es una diseñadora DESconocida (*Luisa NO es conocida como diseñadora*)
- La aparición de cometas es un fenómeno DIScontinuo (*NO es continua la aparición de cometas*)
- Los animales son Amorales (*Los animales NO tienen moral*)
- Los castigos son ANTIpedagógicos (*NO son pedagógicos los castigos*)

- NUNCA me ganó la lotería (*NO me he ganado la lotería*)
- Ricardo JAMÁS miente (*Ricardo NO ha mendido*)
- NINGÚN hombre colombiano usa falda (Los hombres colombianos NO usan falda)*

** Es una afirmación falsa ya que los hombres colombianos de la etnia guambiana si usan falda.*

Un último aspecto a tener en cuenta son los cuantificadores que se utilizan en algunas proposiciones: todos, algunos, ningún, ninguno, sólo, hay, etc. Incluso, proposiciones que no contienen cuantificadores se pueden transformar en proposiciones cuantificadas: "Cada planeta gira sobre su eje" se puede escribir como "todos los planetas giran sobre su eje" (Melo, 2001).

ACTIVIDADES

1. Identificar cuál(es) de las siguientes proposiciones son validas (calificables), explicar por qué son validas o por qué no lo son:

a. *El año 1200 aC. es más reciente que el año 970 de la era Cristiana*
b. *El jugo de lulo tiene muy buen sabor*
c. *La nota máxima en un examen es 10*
d. *Esteban es alto*
e. *Ojalá que no llueva mañana*
f. *¿Podría decirme, por favor, qué hora es?*
g. *Cuatro es mayor que 2*

2. Identificar para cuál(es) de las siguientes proposiciones es muy difícil o imposible establecer con toda certeza si son ciertas o falsas.

a. *Edith Piaf es la alcaldesa de Paris.*
b. *Juan Roa Sierra fue el asesino de Jorge Eliécer Gaitán el 9 de abril de 1948.*
c. *Marco Fidel Suárez fue presidente de Colombia.*
d. *Bogotá es la capital de Bolivia.*

3. Identificar las partes que componen las siguientes proposiciones (sujeto, predicado y la relación entre ambas).

a. *7.0 es menor o igual que 20.5*
b. *El ánguloUno es mayor que 90*
c. *La calificación de Juan Felipe en Historia es menor que 5.0*
d. *Cali y Medellín son ciudades colombianas*
e. *4 y 8 son números menores que 10*
f. *La capital de Colombia es Bogotá*

4. Expresar las siguientes proposiciones en un formato que pueda entender un computador.

a. *7.0 es menor o igual que 20.5*
b. *El ánguloUno es mayor que 90*
c. *El jugo de lulo tiene muy buen sabor*
d. *La calificación de Juan Felipe en Historia es menor que 5.0*
e. *Esteban es alto*
f. *El valor de una calificación no puede ser mayor que 10*
g. *El valor de una calificación no puede exceder a 10*
h. *La capital de Colombia es Bogotá*
i. *4 y 8 son números menores que 10*

EJEMPLO

Supongamos que Mónica quiere ir a comer helado y su padre le propone: "Como hoy entregan tus calificaciones del segundo período, si haz obtenido en matemáticas más de 8.0, vamos a comer helado el próximo sábado, de lo contrario no vamos". La situación "comer helado" está sujeta a la condición "obtener más de 8.0 en matemáticas para el segundo período".

R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Es un problema sencillo de selección doble.
Resultados esperados: Un aviso que indique si el estudiante puede ir a comer helado el próximo sábado o no.
Datos disponibles: La calificación de matemáticas ingresada por el usuario. La regla dice: para ir a comer helado, la nota debe ser mayor que 8.0.
Restricciones: Aplicar la regla dada.
Procesos necesarios: Solicitar al usuario que ingrese la calificación de matemáticas. Evaluar si la calificación es igual o inferior a 8.0; en caso de ser verdadero, reportar "NO come helado"; en caso contrario, reportar "SI come helado".

DISEÑO DEL ALGORITMO

Inicio
Matematicas
Matematicas <= 8.0
VERDADERO
FALSO
NO come helado
SI come helado
Final

TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

```
para helado
  local "matemáticas
  pregunta [Ingrese la calificación de Matemáticas]
  da "matemáticas respuesta
  siotro (o :matemáticas  < 8.0  :matemáticas = 8.0)
       [
        anuncia [NO come helado]
       ]
       [
        anuncia [SI come helado]
       ]
fin
```

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

```
al presionar [bandera]
cambiar el disfraz a [cassy-chillin]
preguntar [Ingrese la calificación de matemáticas] y esperar
fijar [matemáticas] a (respuesta)
si <<(matemáticas) < [8.0]> o <(matemáticas) = [8.0]>>
    cambiar el disfraz a [triste]
    decir [NO come helado   :(] por (5) segundos
si no
    cambiar el disfraz a [feliz]
    decir [SI come helado   :)] por (5) segundos
```

En este ejemplo, la proposición se puede expresar de dos formas equivalentes:

(matemáticas > 8.0)

(matemáticas <= 8.0)

La primera forma es más fácil de manipular por los estudiantes, ya que si la proposición es verdadera entonces "Si come helado" y si la proposición es falsa entonces "NO come helado"; además, utiliza el operador relacional mayor que (>).

La segunda forma (utilizada en el algoritmo) es más compleja. En ella, si la proposición es verdadera entonces "NO come helado" y si la proposición es falsa entonces "SI come helado". En esta forma se presenta un "contrasentido" que puede desorientar a los estudiantes. Además, hay que usar el operador relacional menor o igual que, el cual se traduce en MicroMundos [4] así:

(o :matemáticas < 8.0 :matemáticas = 8.0)

la relación de igualdad no se menciona explícitamente en el enunciado del problema.

EJEMPLO

La profesora Ángela Cristina necesita calcular la nota definitiva para cada uno de los 22 alumnos que asisten a su curso de geometría, con el fin de saber quiénes aprobaron y quiénes reprobaron (para aprobar hay que obtener una nota igual o superior a 6.5). Ella realizó a todos sus estudiantes, en el primer periodo del año lectivo, dos exámenes y asignó un trabajo de investigación. ¿Cómo puedes ayudarle?

R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Se requiere calcular un promedio de tres notas para cada uno de los 22 alumnos.

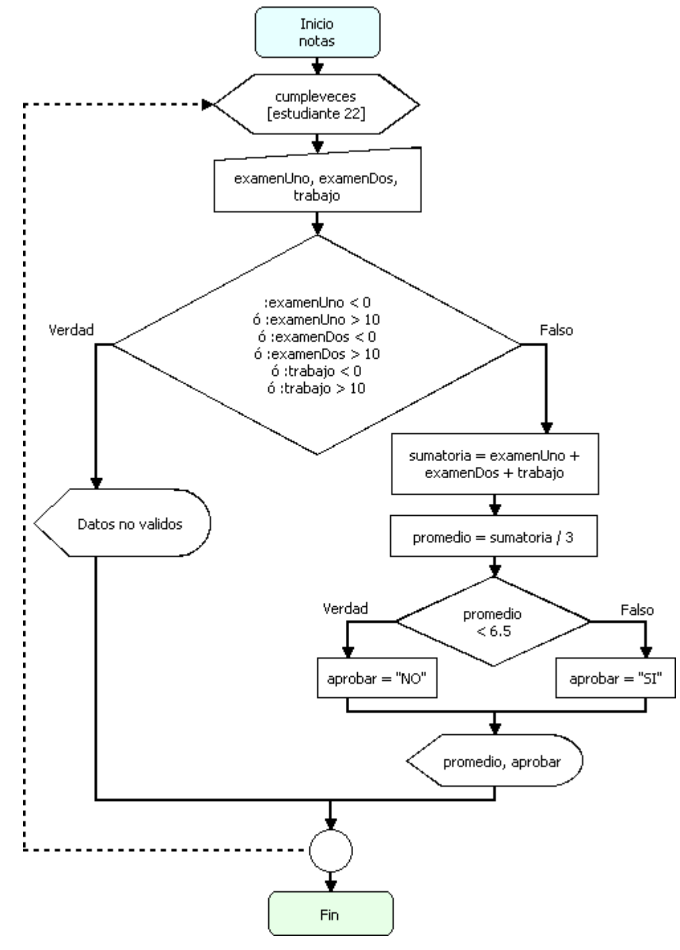
Resultados esperados: La nota definitiva de cada uno de los 22 alumnos y un aviso que indique si aprobó o no.

Datos disponibles: El número de alumnos: 22. Las notas de cada alumno las debe digitar la profesora.

Restricciones: Cada una de las tres notas tienen el mismo porcentaje en la nota definitiva. Tres notas por alumno y 22 alumnos. Todas las notas deben ser mayores o iguales a 1 y menores o iguales a 10. Para aprobar hay que tener un promedio igual o superior a 6.5.

Procesos necesarios: Para cada uno de los 22 alumnos: Leer las tres notas, verificar que estén en el rango permitido (entre 1 y 10), sumarlas, calcular el promedio, verificar si aprobó o no. Mostrar el promedio y un aviso que informe si aprobó o no.

DISEÑO DEL ALGORITMO



TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

```
para notasDos
  cumpleveces [estudiante 22]
   [
     pregunta [Ingrese la nota del primer examen ]
     da "examenUno  respuesta
     pregunta [Ingrese la nota del segundo examen ]
     da "examenDos  respuesta
     pregunta [Ingrese la nota del trabajo de investigación ]
     da "trabajo  respuesta
     siotro (o :examenUno < 1  :examenUno > 10  :examenDos < 1
               :examenDos > 10  :trabajo < 1  :trabajo > 10)
       [
         anuncia [Datos no validos ]
       ]
       [
         da "sumatoria :examenUno + :examenDos + :trabajo
         da "promedio :sumatoria / 3
         siotro (:promedio < 6.5)
           [
             da "aprobar [  ->  NO aprobó  el primer periodo de
Geometría ]
           ]
           [
             da "aprobar [   -> SI aprobó  el primer periodo de
Geometría ]
           ]
```

muestra (frase [El promedio del estudiante] :estudiante + 1
[es]
:promedio :aprobar)
]
]
fin

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

```
al presionar 🏁
fijar estudiante a 0
repetir 22
    fijar estudiante a (estudiante + 1)
    preguntar Ingrese la nota del primer examen y esperar
    fijar examenUno a respuesta
    preguntar Ingrese la nota del segundo examen y esperar
    fijar examenDos a respuesta
    preguntar Ingrese la nota del trabajo de investigación y esperar
    fijar trabajo a respuesta
    si < examenUno < 1 o examenUno > 10 >
        decir Valor del primer examen No valido :(
        esperar 5 segundos
    si no
        si < examenDos < 1 o examenDos > 10 >
            decir Valor del segundo examen No valido :(
            esperar 5 segundos
        si no
            si < trabajo < 1 o trabajo > 10 >
                decir Valor del trabajo No valido :(
                esperar 5 segundos
            si no
                fijar sumatoria a (examenUno + examenDos + trabajo)
                fijar promedio a (sumatoria / 3)
                si < promedio < 6.5 >
                    decir unir Estudiante unir estudiante unir NO aprobó con promedio
                    esperar 5 segundos
                si no
                    decir unir Estudiante unir estudiante unir SI aprobó con promedio
                    esperar 5 segundos
```

Nótese que en este ejemplo se evalúa si la proposición compuesta es verdadera entonces los datos no son validos. Como se utiliza el operador lógico "o", basta con que una de las proposiciones sea verdadera para que toda la proposición compuesta también lo sea. Adicionalmente, en la traducción a Scratch se utilizan estructuras condicionales anidadas (ver la sustentación educativa del uso de estructuras anidadas al final de esta sección).

EJEMPLO

Escribir un procedimiento para leer los valores de dos de los tres ángulos internos de un triángulo y mostrar en pantalla "Es un Triángulo Rectángulo" si efectivamente es un triángulo de este tipo o, en caso contrario, mostrar "No es un Triángulo Rectángulo".

DISEÑO DEL ALGORITMO

para triangulo

Define como locales: ANGULO1, ANGULO2, ANGULO3, EsRectangulo

ANGULO1, ANGULO2

ANGULO3 = 180 - ANGULO1 - ANGULO2
EsRectangulo = NO

ANGULO1=90 — Verdad: EsRectangulo=SI — Falso

ANGULO2=90 — Verdad: EsRectangulo=SI — Falso

ANGULO3=90 — Verdad: EsRectangulo=SI — Falso

EsRectangulo=SI — Verdad: SI es un Triángulo Rectángulo — Falso: NO es un Triángulo Rectángulo

Fin

TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

para triangulo
local "ANGULO1
local "ANGULO2
local "ANGULO3
local "EsRectangulo
pregunta [Ingrese el 1er Ángulo del Triángulo]
da "ANGULO1 respuesta
pregunta [Ingrese el 2do Ángulo del Triángulo]
da "ANGULO2 respuesta

da "ANGULO3 180 - :ANGULO1 - :ANGULO2
da "EsRectangulo "NO ***;inicializa la variable TIPO en NO***

;si uno de los ángulos es igual a 90 cambia el valor de TIPO a SI
si :ANGULO1 = 90 [da "EsRectangulo "SI]
si :ANGULO2 = 90 [da "EsRectangulo "SI]
si :ANGULO3 = 90 [da "EsRectangulo "SI]

;dependiendo del valor de EsRectangulo, muestra que tipo de triángulo es
siotro :EsRectangulo = "SI
[
anuncia [SI es un Triángulo Rectángulo]
]
[
anuncia [NO es un Triángulo Rectángulo]
]
fin

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

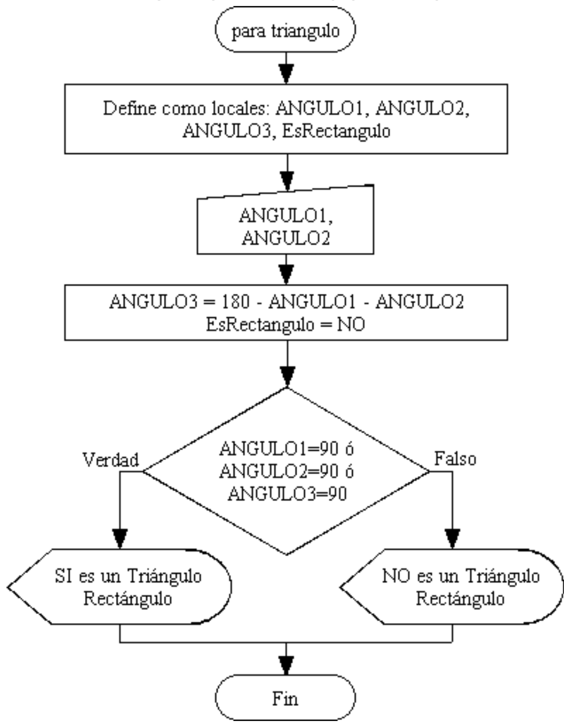
```
al presionar
preguntar [Ingrese el 1er ángulo del triángulo] y esperar
fijar ángulo1 a respuesta
preguntar [Ingrese el 2do ángulo del triángulo] y esperar
fijar ángulo2 a respuesta
fijar ángulo3 a 180 - ángulo1 + ángulo2
fijar esRectángulo a NO
si ángulo1 = 90
  fijar esRectángulo a SI
si ángulo2 = 90
  fijar esRectángulo a SI
si ángulo3 = 90
  fijar esRectángulo a SI
si esRectángulo = SI
  decir [SI es un triángulo rectángulo!!!!]
  esperar 5 segundos
si no
  decir [NO es un triángulo rectángulo :(]
  esperar 5 segundos
```

En este ejemplo se evalúa una a una las proposiciones para determinar sí uno de los ángulos es igual a 90. Nótese que la variable esRectángulo se inicializa con el valor "NO", en caso de que cualquiera de los ángulos sea igual a 90, entonces la variable esRectangulo se cambia a "SI". Finalmente se evalúa el valor resultante de esta variable para mostrar el mensaje "si" o "no" es un triángulo rectángulo.

EJEMPLO

Escribir un procedimiento para leer los valores de dos de los tres ángulos internos de un triángulo y mostrar en pantalla "Es un Triángulo Rectángulo" si efectivamente es un triángulo de este tipo o, en caso contrario, mostrar "No es un Triángulo Rectángulo". Utilizar operadores lógicos en la solución.

DISEÑO DEL ALGORITMO



TRADUCCIÓN DEL ALGORITMO EN MICROMUNDOS

```
para triangulo
  local "ANGULO1
  local "ANGULO2
  local "ANGULO3
  pregunta [Ingrese el 1er Ángulo del Triángulo]
  da "ANGULO1 respuesta
  pregunta [Ingrese el 2do Ángulo del Triángulo]
  da "ANGULO2 respuesta
  da "ANGULO3 180 - :ANGULO1 - :ANGULO2
  siotro (o :ANGULO1 = 90  :ANGULO2 = 90  :ANGULO3 = 90)
    [
      anuncia [SI es un Triángulo Rectángulo]
    ]
    [
      anuncia [NO es un Triángulo Rectángulo]
    ]
fin
```

TRADUCCIÓN DEL ALGORITMO EN SCRATCH

```
al presionar
preguntar [Ingrese el 1er ángulo del triángulo] y esperar
fijar ángulo1 a respuesta
preguntar [Ingrese el 2do ángulo del triángulo] y esperar
fijar ángulo2 a respuesta
fijar ángulo3 a 180 - ángulo1 + ángulo2
si ángulo1 = 90 o ángulo2 = 90 o ángulo3 = 90
  decir [SI es un triángulo rectángulo!!!!]
  esperar 5 segundos
si no
  decir [NO es un triángulo rectángulo :(]
  esperar 5 segundos
```

Esta es una solución más simple, corta y elegante que la hallada en el ejemplo anterior. La utilización del operador lógico "o" permite evaluar en una sola instrucción si alguno de los ángulos vale 90. Un problema inesperado a plantear a los estudiantes consiste en preguntarles ¿qué pasa si alguien digita valores para los ángulos que sumados den más de 180? ¿cómo controlar que esto no suceda?

TIP
En un programa se pueden incluir varias cláusulas condicionales anidadas. Los comandos si y siotro se pueden anidar de igual manera como se anida la función si (if) en la Hoja de Cálculo.

ACTIVIDAD
Tomando como base el ejemplo anterior, realizar las modificaciones necesarias para que adicionalmente, el programa muestre en pantalla "No es un Triángulo" en caso de que cualquiera de los ángulos sea menor o igual a cero.

Estructuras condicionales anidadas

Hay situaciones que requieren el uso de estructuras condicionales anidadas. En estas, el resultado de la primera proposición implica evaluar a continuación una segunda proposición y esta a su vez requiere que se evalúe una tercera proposición, y así sucesivamente, hasta agotar todas las condiciones.

Plantear estructuras algorítmicas anidables (iterativa y condicional) requiere procesos de pensamiento asociados con el sistema operatorio de clasificación o inclusión. Este sistema se empieza a adquirir cuando los niños trabajan con inclusiones de clases tales como: gatos(A) < felinos(B) < animales(C) < seres vivos(D). Pero el sistema de clasificación está basado en cinco operaciones: Composición (A+A'=B, B+B'=C, B-A'=A, C-B'=B); Inversión (-A-A'=-B); Identidad (A-A=0); Tautología (A+A=A, donde A+B=B); Asociatividad (A+(A'+B')=(A+A')+B').

Para poder determinar rápida y efectivamente cuándo la solución de un problema requiere estructuras anidadas, se deben manejar sistemas de clasificación o inclusión de clases.

El siguiente caso ilustra muy bien este punto: "Se requiere elaborar un procedimiento que permita determinar para un grupo de 10 personas si tienen derecho o no a jubilarse a partir de los datos género, edad y años de aportes; y las siguientes condiciones: si es hombre debe tener más de 65 años de edad y más de 60 años si es mujer, pero en todo caso se deben haber realizado aportes por más de 20 años".

Para resolver este problema se puede plantear la siguiente inclusión de clases: edad(A) < género(B) < aportes(C).

```
al presionar [bandera]
repetir 10
    preguntar ¿Cuántos años lleva aportando? y esperar
    fijar aportes a respuesta
    preguntar ¿Qué edad tienen actualmente? y esperar
    fijar edad a respuesta
    preguntar H=Hombre / M=Mujer y esperar
    fijar género a respuesta
si aportes > 20
    si género = M
        si edad > 60
            decir SI TIENE DERECHO A JUBILARSE
            esperar 5 segundos
        si no
            decir NO TIENE DERECHO A JUBILARSE
            esperar 5 segundos
    si género = H
        si edad > 65
            decir SI TIENE DERECHO A JUBILARSE
            esperar 5 segundos
        si no
            decir NO TIENE DERECHO A JUBILARSE
            esperar 5 segundos
si no
    decir NO TIENE DERECHO A JUBILARSE
    esperar 5 segundos
```

.

UNIDAD 4: DEPURACIÓN

CUANDO SE PRESENTAN PROBLEMAS

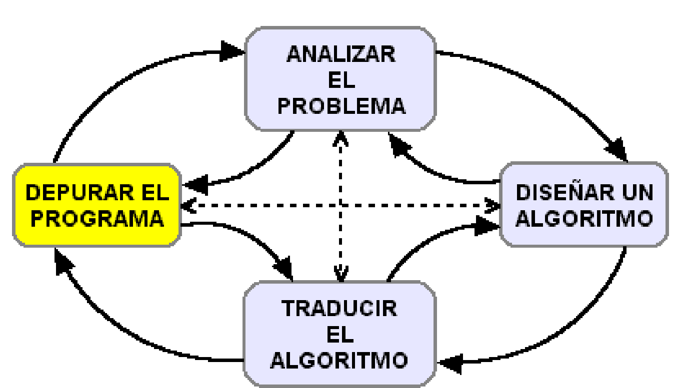


Ilustración 4-1: Cuarta fase del ciclo de programación.

Es muy difícil elaborar procedimientos perfectos en un primer intento y la dificultad aumenta a medida que los problemas se vuelven más complejos. Después de traducir el algoritmo en un lenguaje de programación, el procedimiento resultante debe ser probado y los resultados validados (revisión). A este proceso se le conoce como depuración y contribuye a mejorar en los estudiantes la capacidad para resolver problemas puesto que la depuración basada en la retroalimentación es una habilidad útil para toda la vida (Stager, 2003).

La depuración de un procedimiento hace parte fundamental del ciclo de programación y desde el punto de vista educativo estimula en los estudiantes la curiosidad, la perspectiva, la comunicación y promueve valores como responsabilidad, fortaleza, laboriosidad, paciencia y perseverancia. La programación facilita un diálogo interior en el cual la retroalimentación constante y el éxito gradual empujan a los alumnos a ir más allá de sus expectativas (Stager, 2003).

Otras dos actividades relacionadas con esta etapa, que no se tratarán en esta guía, son la afinación y la documentación. La primera consiste en realizar retoques para lograr una mejor apariencia del programa (en pantalla o en los resultados impresos) o para ofrecer funcionalidades más allá de los resultados esperados, especificados en la fase de análisis del problema. La segunda tiene un carácter eminentemente comunicativo, con la documentación de un programa se pone a prueba la capacidad del estudiante para informar a otras personas qué hace su programa, cómo lo hace y el significado de cada elemento utilizado. Esta actividad se puede llevar a cabo mediante comentarios introducidos al código o por medio de documentación formal en un documento que se anexa al procedimiento elaborado.

Depuración

La corrección de fallas es una de las situaciones que mayor frecuencia tienen en el mundo profesional. Con esta actividad se intenta identificar fallas sintácticas o lógicas en programas que no funcionan adecuadamente; una vez aislada la falla, esta se soluciona y se vuelve a probar el programa y a validar los resultados. Según Jonassen (2003), para corregir fallas efectiva y eficientemente se requiere conocimiento del sistema (comprensión conceptual de cómo funciona el sistema), conocimiento procedimental (cómo llevar a cabo tanto procedimientos de solución de fallas, como actividades de prueba) y conocimiento estratégico (saber cuándo, dónde y por qué aplicar procedimientos de solución de fallas y actividades de prueba).

Fallas de sintaxis

Este tipo de fallas solo se presenta en MicroMundos ya que el entorno de programación de Scratch es gráfico y los estudiantes no deben escribir el código. Además, los bloques con las instrucciones son autoencajables, por tanto no es posible ubicar un bloque de manera que se generen fallas de sintaxis.

Las fallas sintácticas son las más sencillas de identificar ya que el entorno de programación indica dónde se ha producido el error y de que tipo es. Por ejemplo, MicroMundos reportará el error "valorDos no tiene valor en prueba" cuando, en el procedimiento "prueba", se intenta mostrar el contenido de la variable "valorDos" sin haberla asignado antes.

En caso de presentarse una falla de sintaxis, el estudiante debe:

- Comprender el mensaje de error que reporta el ambiente de programación (apoyarse en las opciones de ayuda que ofrece MicroMundos o en el docente).
- Examinar el código del programa para identificar en cuál instrucción se encuentra la falla
- Corregir la falla
- Probar el programa de nuevo

EJEMPLO 4-1

Retomemos el ejemplo del estudiante que aprueba un examen cuando obtiene una calificación mayor o igual a seis (ejemplo 3-14). Se requiere elaborar un procedimiento que pida al usuario una calificación, aplique el criterio de aprobación e imprima "Aprobado" o "Reprobado", según sea el caso.

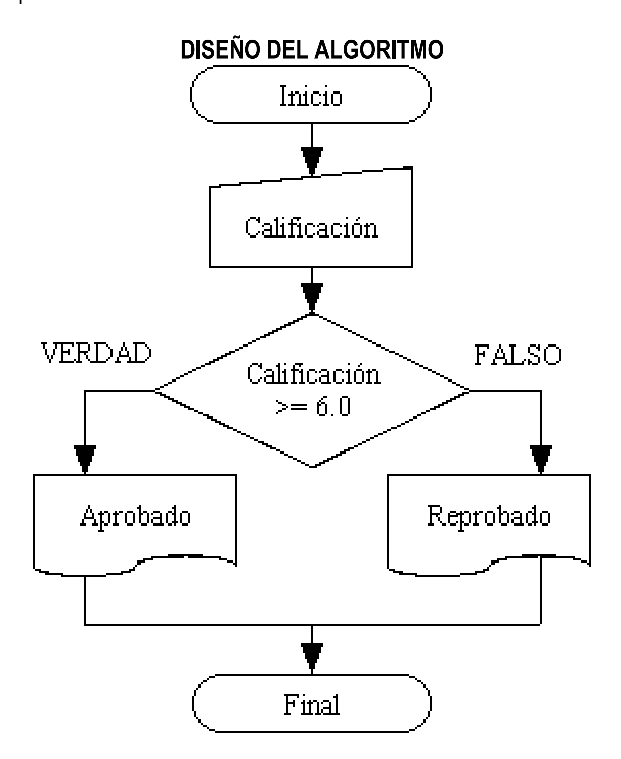
R/.

ANÁLISIS DEL PROBLEMA

Formular el problema: Es un problema sencillo de selección doble.

Resultados esperados: Un aviso que reporte si el estudiante “Aprobó” o “Reprobó” el examen.
Datos disponibles: La calificación ingresada por el usuario. Para aprobar, la nota debe ser mayor o igual a 6.0.
Restricciones: Aplicar el criterio de aprobación.
Procesos necesarios: Solicitar al usuario que ingrese la calificación. Evaluar si la calificación es igual o superior a 6.0; en caso de ser verdadero, reportar “Aprobado”; en caso contrario, reportar “Reprobado”.

DISEÑO DEL ALGORITMO



TRADUCCIÓN DEL ALGORITMO

```
para aprueba1
  local "calificación
  preguntas [Ingrese la Calificación]
  da "calificación respuesta
  siotro (o :calificación > 6.0  :calificación = 6.0)
      [
       anuncia [Aprobado]
      ]
      [
       anuncia [Reprobado]
     ]
fin
```

Obsérvese que MicroMundos arroja un mensaje de error cuando se ejecuta el procedimiento: “*No sé cómo hacer preguntas en aprueba1*”. Este error se produce porque el comando ***pregunta*** está mal escrito, en su lugar se escribió ***preguntas***. El problema se soluciona al escribir correctamente el comando indicado.

Fallas de lógica

Este tipo de falla se presenta tanto en MicroMundos como en Scratch. Para identificar fallas de tipo lógico en un procedimiento que no se interrumpe en ningún momento y que arroja unos resultados, estos se deben revisar cuidadosamente. Uno de los procedimientos más utilizados es el que se conoce como prueba de escritorio. Esta consiste seguir paso a paso cada una de las instrucciones del procedimiento, asignando valores iniciales a variables y constantes y, realizando las operaciones indicadas en cada instrucción hasta llegar al final del procedimiento. Luego, comparar los resultados que produce la prueba de escritorio con los resultados que arroja el procedimiento; ambos conjuntos de resultados deben ser iguales. Esta metodología es viable cuando los procedimientos son sencillos, con pocas instrucciones.

En caso de presentarse una falla de lógica, el estudiante debe:

- Examinar el diagrama de flujo para detectar instrucciones que faltan, sobran o que se encuentran en la posición incorrecta
- Asegurarse que las instrucciones representan rigurosamente el diagrama de flujo
- Utilizar las opciones de MicroMundos y Scratch para ver la ejecución del programa en forma lenta.

EJEMPLO 4-2
Continuamos con el mismo ejemplo: Elaborar un procedimiento que pida al usuario una calificación, aplique el criterio de aprobación e imprima “Aprobado” o “Reprobado”, según sea el caso. Con el fin de ejemplificar una falla de lógica, se debe digitar en el área de procedimientos de MMP el siguiente código que representa el algoritmo del ejemplo 4-1.

TRADUCCIÓN DEL ALGORITMO

```
para aprueba2
  local "calificación
  pregunta [Ingrese la Calificación]
  da "calificación respuesta
  siotro (o :calificación < 6.0  :calificación = 6.0)
       [
        anuncia [Aprobado]
       ]
       [
        anuncia [Reprobado]
     ]
fin
```

Obsérvese que el procedimiento “aprueba2” se ejecuta correctamente; no aparece ningún error de sintaxis. Pero no funciona bien desde el punto de vista lógico. Si lo probamos con el valor 5, el procedimiento nos reporta que el estudiante aprobó la materia, cuando esto no es correcto. Si lo probamos con 7, nos reporta que el estudiante reprobó, cuando tampoco es exacto. En cambio, cuando lo probamos con 6 nos dice que el estudiante aprobó y este dato si es correcto. El problema radica en que en el comando *siotro* se digitó erróneamente “<” en lugar de “>”, tal como aparece en el diagrama de flujo.

Adicionalmente, en este ejemplo se puede observar la sintaxis en MicroMundos de los operadores lógicos (y, o,

no), mediante los cuales se unen proposiciones sencillas para construir proposiciones compuestas. Estos deben ir en seguida del paréntesis que abre la proposición:
siotro (o :calificación > 6.0 :calificación = 6.0)
La proposición se lee así:
"calificación mayor que 6.0 o calificación igual a 6.0".

En la detección y eliminación de fallas en un procedimiento, cuenta mucho la cantidad de fallas similares que el estudiante ha tenido oportunidad de resolver. La "experiencia" es un factor crucial; incluso, las fallas que se recuerdan con mayor precisión son aquellas cuya solución presentó mayor dificultad (Jonassen, 2003). Con la depuración, el estudiante realiza un conjunto de actividades que contribuyen a aprender de ese problema, a reinterpretarlo, a formular otros nuevos y a incrementar la comprensión de la solución hallada. Esta incita a los estudiantes a preguntarse: "¿puedo obtener el mismo resultado de una forma diferente?" y "¿puedo utilizar los métodos empleados para solucionar este problema en la solución de otros que se me han presentado antes?"

ANEXO 1

RESUMEN DE COMANDOS DE MICROMUNDOS Y SCRATCH

A continuación se ofrece un resumen de las primitivas de MicroMundos y Scratch utilizadas en esta Guía.

DESCRIPCIÓN	MICROMUNDOS	SCRATCH
ADELANTE Mueve la tortuga hacia adelante. Los valores mínimos y máximos para adelante son -9999 y 9999, respectivamente.	Adelante (ad) número Ver at, de, iz. *adelante control1* *repite 4 [ad 50 de 90]*	mover 10 pasos
ALTO Detiene el procedimiento que está activo. Alto solo puede usarse dentro de un procedimiento.	Alto Ver deténtodo, deténme y reporta. *para contar :número* *si :número > 100 [alto]* *muestra :número* *contar :número + 5* *fin*	detener programa detener todo
ANUNCIA Muestra el mensaje en una caja de alerta. Haciendo clic en Aceptar se cierra la caja. Si mueve la caja de alerta a una nueva posición mientras está en la pantalla, esta será la posición en que la próxima caja de alerta aparecerá en el proyecto.	anuncia palabra-o-lista *anuncia "bienvenido* *anuncia [Hola]* *anuncia texto1* Ver pregunta y respuesta.	decir ¡Hola! decir ¡Hola! por 2 segundos
ATRÁS Mueve la tortuga hacia atrás. Los valores mínimos y máximos para atrás son -9999 y 9999, respectivamente.	atrás (at) número Ver ad, de, iz *atrás 45* *at control1* *repite 4 [at 50 iz 90]*	mover -10 pasos
AZAR Devuelve un número entero positivo (incluyendo el 0) menor que número. El número máximo es 9999.	azar número *azar control1* *azar 2* *Repite 26 [ad azar 30 de azar 60]*	número al azar entre 1 y 10
BNOMBRES Borra de la memoria todas las variables globales. MicroMundos no borra las variables cuando se abre o se crea un nuevo proyecto. Por lo tanto se recomienda usar bnombres cada vez que se inicie un nuevo proyecto.	bnombres Ver nombres. *bnombres*	No aplica
CON PLUMA Pone la pluma a la tortuga en uso. La tortuga dejará una marca cuando se mueva, pero no cuando sea arrastrada.	Cp Ver sp. *repite 6 [sp ad 10 cp ad 10]*	bajar lápiz
CUMPLEVECES Activa la lista de instrucción para cada uno de los valores especificados en la serie. La primera entrada es una lista con un nombre de variable temporal y un número máximo. La segunda entrada es una lista de instrucciones que usa la variable incluida en la primera lista.	cumpleveces serie lista-de-instrucción Ver cumplelista. *cumpleveces [i 8][muestra :i]* *cumpleveces [i 360] [fcolor :i / 10 ad 40 at 40 de 2]*	No aplica
DA (ASIGNA) Crea una variable y le asigna el valor palabra-o-lista. Estas variables mantienen su valor siempre y cuando no se las borre o se cierre MicroMundos.	da vpalabra palabra-o-lista Ver nombra, cosa, bnombre, nombres y creavarproyecto. *da "equipo [t1 t2 t3]* *da "texto texto1*	fijar variable a 0

DESCRIPCIÓN	MICROMUNDOS	SCRATCH
DERECHA Gira la tortuga hacia la derecha. El máximo valor para derecha es 9999.	de número; derecha número Ver ad, at,iz. *derecha 45* *repite 5 [de -90 ad 15 de 90 ad 15]*	girar ↻ 15 grados
ESCRIBE Escribe la palabra o la lista en la caja de texto en uso. El texto va seguido de un retorno de línea y de una secuencia de avance de línea.	escribe (es) palabra-o-lista Ver inserta. *escribe "holaescribe [Bienvenidos de nuevo]* *escribe texto1*	No aplica
ESPERA Origina una pausa en la ejecución de un programa o de una instrucción. El tiempo se mide en décimas de segundo.	espera número *espera 2* *espera control1*	esperar 1 segundos
ESPERAHASTA Antes de activar otra instrucción, espera hasta que cierto-o-falso-lista-de-instrucciones devuelva cierto. La entrada debe ser una lista de instrucciones que devuelva cierto o falso cuando se active.	esperahasta cierto-o-falso-lista-de-instrucciones *esperahasta [tocando? "t1 "t2] t3, mt*	esperar hasta que
FRASE Devuelve una lista formada por sus entradas (palabras o listas). Frase puede tomar más de dos entradas cuando frase y sus entradas se encierran entre paréntesis. Frase no mostrará los corchetes de las listas individuales dadas como entrada.	frase (fr) palabra-o-lista1 palabra-o-lista2 (frase palabra-o-lista1 palabra-o-lista2 palabra-o-lista3...) Ver lista. *frase "uno [dos]* *(frase "una "gran [ciudad])* *frase texto1 texto2*	decir ¡Hola! unir hola mundo
RUMBO Fija el rumbo de la tortuga (en grados) en la dirección especificada. Los grados corresponden a los de la brújula: 0 apunta al norte, 90 al este, 180 al sur y 270 al oeste. Derecha e izquierda hacen girar a la tortuga un determinado número de grados partiendo de la posición en que se encuentra en ese momento. En cambio, frumbo hace que la tortuga apunte en una dirección específica, independientemente de su orientación anterior.	frumbo número Ver rumbo. *frumbo 0* *frumbo control1*	apuntar en dirección 90
IZQUIERDA Gira la tortuga a la izquierda. El máximo valor para izquierda es 9999.	izquierda (iz) número Ver ad, at, de. *iz 45* *ad 15* *repite 4 [iz -90 ad 15 iz 90 ad 15]*	girar ↺ 15 grados
LIMPIA PANTALLA Limpia los gráficos sin cambiar la posición de la tortuga.	Limpia Ver bg y congelaf. *limpia*	borrar
LOCAL (VARIABLES) Crea una variable local en el procedimiento dentro del cual se utilizó el mando local. Local sólo puede ser usado dentro de un procedimiento.	local palabra-o-lista Ver asigna, da y nombra. *local "lista* *da "lista páginas*	No aplica
MUESTRA Escribe una palabra o una lista en el Centro de Mando. Los corchetes exteriores de la lista no aparecen.	muestra palabra-o-lista *muestra "hola* *muestra [hola todos]* *muestra :texto1*	decir ¡Hola! unir hola mundo

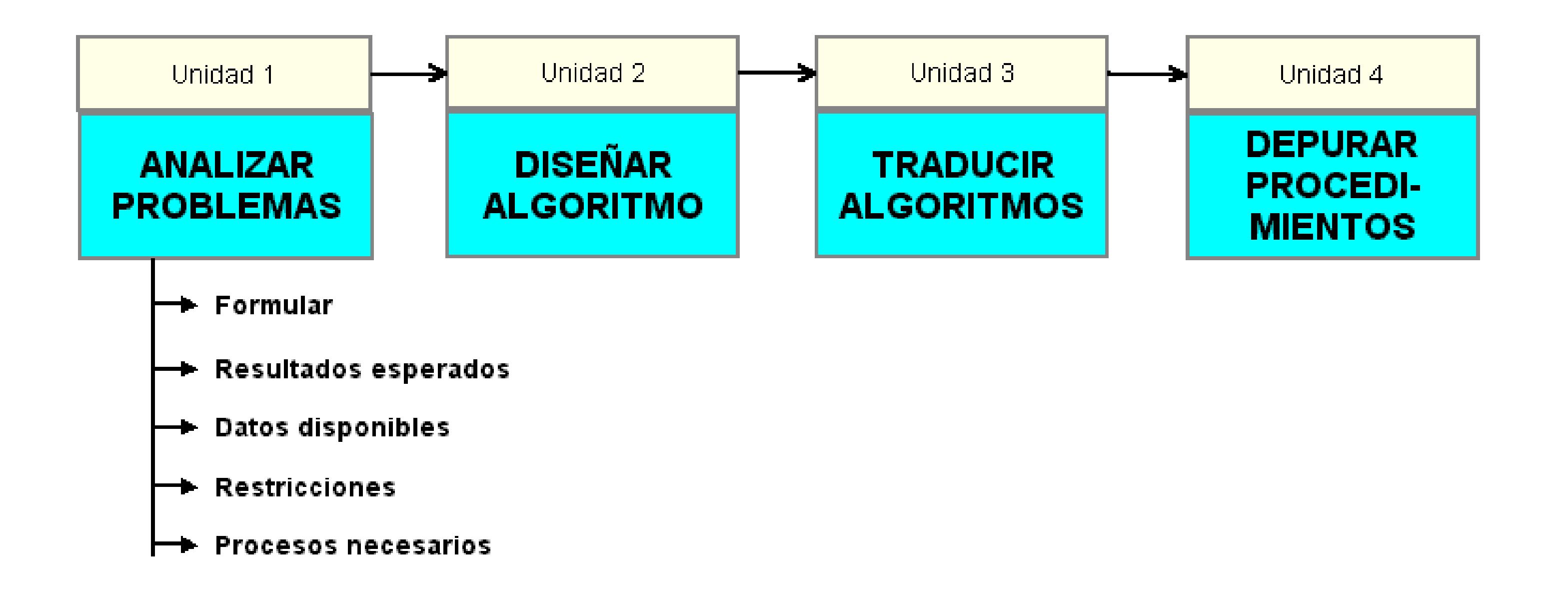
DESCRIPCIÓN	MICROMUNDOS	SCRATCH
NO Devuelve el valor lógico inverso de su entrada.	no cierto-o-falso Ver "y" y "o". *no 1 = control1*	No aplica
NOMBRES Devuelve los nombres de todas las variables con sus valores.	Nombres Ver bnombres *muestra nombres*	mostrar variable Puntaje
O Devuelve cierto si alguna de sus entradas devuelve cierto. Si se usan más de dos entradas, o y sus entradas deben estar entre paréntesis.	o cierto-o-falso1 cierto-o-falso2 (o cierto-o-falso1 cierto-o-falso2 cierto-o-falso3...) Ver "y" y "no". *o 1 = control1 control1 < 4* *(o 1 = control1 control1 < 4 2 > control2)*	o
PORSIEMPRE Activa la entrada en forma repetitiva como un proceso paralelo independiente. Use detén, Detener Todo, o Ctrl+Inter para detener el proceso.	porsiempre palabra-o-lista-a-activar Ver lanza, y Tiempo y Sincronización en los Temas de Ayuda de MicroMundos. *porsiempre [ad 2]* *porsiempre "cuar* *porsiempre texto1*	por siempre
PREGUNTA Abre una caja de diálogo que muestra la pregunta y una zona para escribir la respuesta. Respuesta devuelve lo que se ha escrito en la caja de diálogo. Si se escribe una pregunta muy larga, solo aparecerá la parte que quepa dentro del espacio dado. Si se arrastra la caja de diálogo a una nueva posición mientras ésta muestra la pregunta, ésta será la posición en que la nueva caja de diálogo se abrirá dentro del proyecto.	pregunta palabra-o-lista Ver respuesta. *pregunta "¿Bien?* *pregunta [¿Cómo estás?]* *pregunta texto1*	preguntar Cuál es su nombre? y esperar respuesta
REPITE Activa la lista de instrucciones el número de veces especificado.	repite número [lista-de-instrucciones] Ver cumpleveces y cumplelista para opciones más avanzadas. *repite 96 [at 40 ad 40 de 4]*	repetir 10
RESPUESTA Devuelve el contenido de la última respuesta que se escribió en la caja de diálogos de pregunta. Usando pregunta y respuesta, se pueden utilizar las palabras escritas en el teclado para crear un programa interactivo.	respuesta *da "valor respuesta* *si respuesta = "sí [ffig "contento]* *si respuesta = "no [ffig "triste]*	preguntar Cuál es su nombre? y esperar respuesta
SI Activa la lista de instrucciones sólo si la proposición de la primera entrada devuelve cierto.	si cierto-o-falso [lista-a-activar] Ver siotro. *si 5 > 2 [anuncia "correcto]*	si
SIOTRO Activa la primera lista de instrucciones si la proposición devuelve cierto. Activa la segunda lista de instrucciones si la proposición devuelve falso.	siotro cierto-o-falso [lista-de-instrucciones1] [lista-de-instrucciones2] Ver si. *siotro colordebajo = 15 [ad 50] [at 50]*	si si no
SIN PLUMA Significa sin pluma. Saca la pluma a la tortuga en uso. La tortuga no dejará ningún trazo cuando se mueva.	Sp Ver cp. *repite 6 [sp ad 10 cp ad 10]*	subir lápiz

DESCRIPCIÓN	MICROMUNDOS	SCRATCH
Y Devuelve cierto si todas sus entradas devuelven cierto. Si se utilizan más de dos entradas, y debe estar entre paréntesis junto con todas sus entradas.	y cierto-o-falso1 cierto-o-falso2 (y cierto-o-falso1 cierto-o-falso2 cierto-o-falso3...) Ver “o” y “no”. *y 1 = 1 3 < 4* *(y 1 = 1 3 < 4 2 > 1)*	

ANEXO 2

ESQUEMA DE LA PRESENTE GUÍA

Esta guía está escrita para docentes; por tanto, los temas se presentan en una secuencia propia de la programación de computadores, tal como la establecen y manejan los especialistas de esta disciplina. Esta manera de exponer los contenidos que se deben enseñar, facilita al docente la consulta de cualquier tema. Además, es la forma como por lo general se presentan los libros de texto utilizados por los estudiantes universitarios.



ANEXO 3

PLAN DE TRABAJO CON ESTUDIANTES

La experiencia de utilizar esta guía con estudiantes de grado 5º de básica primaria en el Instituto Nuestra Señora de la Asunción (INSA – http://www.insa-col.org) demostró que la secuencia óptima de presentación a los estudiantes de los temas de Algoritmos y programación, para su aprendizaje, debe ser diferente al orden en que se exponen en esta guía. De la misma forma como Onrubia & Rochera & Barberà (2001) aconsejan secuenciar la enseñanza de las matemáticas, la sucesión de contenidos de la programación se debe planear de acuerdo a una estructura helicoidal, en la que los distintos temas se retomen en diversas ocasiones a lo largo del proceso de enseñanza y aprendizaje, de forma que el estudiante pueda comprender e interiorizar progresivamente dichos contenidos. Adicionalmente, este tipo de estructura facilita el establecimiento de relaciones cada vez más elaboradas entre los distintos aspectos de los algoritmos y la programación.
La secuencia que se propone en el cuadro siguiente evita que la primera fase del ciclo de programación (analizar el problema) se convierta en algo pesado y tedioso para los estudiantes. Cada unidad se puede cubrir en uno de los periodos académicos en los que generalmente se divide el año lectivo.

CICLO DE PROGRAMACIÓN	UNIDAD 1 (1er Período)	UNIDAD 2 (2do Período)	UNIDAD 3 (3er Período)	UNIDAD 4 (4to Período)
ANALIZAR EL PROBLEMA → Formular el problema → Resultados esperados → Datos disponibles → Restricciones → Procesos necesarios	• Problemas matemáticos • Ejemplos • Actividades • Ciclo de programación	• Ciclo de programación • Análisis de problemas (formular el problema; resultados esperados; datos disponibles; restricciones; procesos necesarios)	• Análisis de problemas (formular el problema; resultados esperados; datos disponibles; restricciones; procesos necesarios)	• Análisis de problemas (formular el problema; resultados esperados; datos disponibles; restricciones; procesos necesarios)
DISEÑAR UN ALGORITMO	• Qué es un algoritmo • Pseudocódigo • Identificadores • Variables • Constantes • Diagrama de flujo (símb.) • Pensamiento algorítmico	• Pensamiento algorítmico • Variables • Constantes • Operadores • Expresiones • Diagrama de flujo (elaboración) • Uso de software de AV (elaboración)	• Diagrama de flujo (elaboración) • Expresiones	• Diagrama de flujo (elaboración) • Tipos de datos
TRADUCIR EL ALGORITMO	• Introducción a la prog. • Procedimientos • Palabras reservadas • Comentarios • Procesos • Interactividad usuario	• Procedimientos • Estructura secuencial	• Estructura de repetición	• Estructura condicional
DEPURAR EL PROGRAMA	• Tipos de fallas • Fallas de sintaxis • Fallas de precaución	• Fallas de sintaxis • Fallas de concepción • Fallas de lógica • Prueba de escritorio	• Fallas de lógica • Prueba de escritorio • Verificación de resultados	• Fallas de lógica • Prueba de escritorio • Verificación de resultados • Documentación • Afinamiento
TEMAS DE MATEMÁTICAS	*Análisis de problemas* • Solución de problemas	*Variables y constantes* • Áreas • Perímetros *Estructura secuencial* • Operaciones • Procedimientos	*Operadores y expresiones* • Polinomios aritméticos *Estructura de repetición* • Multiplicación • Potenciación • Polígonos • Círculos	*Expresiones* • Polinomios aritméticos *Estructura condicional* • Proposiciones • Relaciones de orden

ANEXO 4

PROGRAMACIÓN DE COMPUTADORES
UNA PROPUESTA DE CURRÍCULO PARA SCRATCH

Esta propuesta hace parte del "Modelo Curricular Interactivo de Informática"
http://www.eduteka.org/modulos.php?catx=9&idSubX=280&ida=937&art=1
http://www.eduteka.org/curriculo2/Herramientas.php?codMat=16

Definición
Scratch es un entorno de programación desarrollado por un grupo de investigadores del Lifelong Kindergarten Group del Laboratorio de Medios del MIT, bajo la dirección del Dr. Mitchel Resnick.
Este entorno aprovecha los avances en diseño de interfaces para hacer que la programación sea más atractiva y accesible para todo aquel que se enfrente por primera vez a aprender a programar. Según sus creadores, fue diseñado como medio de expresión para ayudar a niños y jóvenes a expresar sus ideas de forma creativa, al tiempo que desarrollan habilidades de pensamiento lógico y de aprendizaje del Siglo XXI, a medida que sus maestros superan modelos de educación tradicional en los que utilizan las TIC simplemente para reproducir prácticas educativas obsoletas.

Alcance
- Se busca que el estudiante utilice tanto estructuras de control como el conjunto de instrucciones (bloques) que ofrece el entorno de programación Scratch para elaborar procedimientos con el fin de solucionar problemas, elaborar simulaciones o comunicar información. Los estudiantes no elaborarán programas complejos, sólo se concentrarán en la elaboración de procedimientos.

Objetivo General
- Al terminar la instrucción, el estudiante estará en capacidad de actuar creativamente para elaborar programas en Scratch que resuelvan situaciones planteadas por el docente tales como: historias interactivas, simulaciones y solución de problemas.

Objetivos Específicos
- Utilizar el sitio Web de Scratch y registrarse (Actividad 0)
- Reconocer el entorno de trabajo de Scratch (actividad 0)
- Utilizar apropiadamente las funciones básicas del entorno de trabajo de Scratch (abrir y cerrar programa, abrir y cerrar proyectos existentes, cambiar el lenguaje del entorno) (actividad 0)
- Reconocer el entorno de trabajo del editor de pinturas (actividad 0)
- Utilizar el editor de pinturas (Actividad 1)
- Crear y editar objetos, disfraces, fondos y escenario (Actividad 1)
- Dar instrucciones básicas a objetos (al presionar, por siempre, esperar, mover, etc) (actividad 1)
- Crear historias interactivas con Scratch incorporando instrucciones como: pensar, pensar por N segundos, decir, decir por N segundos, cambiar disfraz e instrucciones de sonido (actividad 2)
- Explicar en sus propias palabras qué es un evento y qué es un hilo (actividad 3)
- Crear programas que manejen eventos (sensores) y multihilos (enviar a todos, al recibir, al presionar objeto, mostrar, esconder) (actividad 3 y actividad 4)
- Reconocer las formas de documentar la funcionalidad de un Proyecto en Scratch (agregar comentarios, notas del proyecto). (actividad 4)
- Realizar modificaciones a programas existentes para mejorarlos. (actividad 4)
- Elaborar dibujos mediante el movimiento de objetos (funcionalidades de Lápiz) (Actividad 4B)
- Utilizar operaciones matemáticas y booleanas
- Crear y utilizar variables y listas
- Compartir con otras personas los trabajos realizados en Scratch.

CONTENIDOS

Utilizar el sitio web de Scratch y registrarse
- Acceder al sitio Web de Scratch
- Explorar el sitio Web de Scratch
- Registrarse en el sitio Web de Scratch

Evaluación (Indicadores de logro)
- Accede a la página Web de Scratch y se registra en ella.

Períodos de clase: 1
Actividades:

TEMA: Actividad Introductoria a Scratch http://www.eduteka.org/proyectos.php/5/2483

Reconocer el entorno de trabajo de Scratch

- Reconocer/Identificar la Barra de Títulos
- Reconocer la Barra de Menús
- Reconocer Bandera Verde y el Botón Parar
- Reconocer el Escenario
- Reconocer la información de Coordenadas del Ratón dentro del Escenario
- Reconocer el Modo de Presentación
- Reconocer los Botones de Objeto
- Reconocer la Lista de Objetos
- Reconocer el Área de Información del Objeto
- Reconocer el Área de Programa
- Reconocer el Área de Disfraces
- Reconocer el Área de Sonidos
- Reconocer el Área de Fondos
- Reconocer la Paleta de Bloques

Evaluación (logros)

- Sin ayuda de referencias, explica en sus propias palabras las principales partes del entorno de trabajo de Scratch

Períodos de clase: 1
Actividades:
TEMA: Actividad Introductoria a Scratch http://www.eduteka.org/proyectos.php/5/2483

Utilizar apropiadamente las funciones básicas del entorno de trabajo de Scratch (abrir y cerrar programa, abrir y cerrar proyectos existentes, cambiar el lenguaje del entorno)

- Abrir y cerrar el programa
- Abrir y cerrar un proyecto existente
- Crear un proyecto nuevo
- Importar un Objeto Sorpresa
- Duplicar, borrar, agrandar y achicar objeto (Barra herramientas)
- Guardar un proyecto
- Seleccionar lenguaje (idioma) de la interfaz
- Ejecutar un proyecto utilizando el botón Bandera Verde
- Detener la ejecución de un programa utilizando el botón Parar Todo
- Seleccionar el modo presentación

Evaluación (logros)

- El estudiante demuestra que utiliza las opciones básicas del entorno, al: abrir, ejecutar y cerrar, proyectos existentes;
- Crea un proyecto nuevo en el que incorpore al menos un Objeto, le cambie el idioma a la interfaz y lo ejecute en los diferentes modos de presentación que permite el entorno de trabajo.

Períodos de clase: 2
Actividades:
TEMA: Actividad Introductoria a Scratch http://www.eduteka.org/proyectos.php/5/2483

Reconocer el entorno de trabajo del editor de pinturas

- Reconocer la opción Importar
- Reconocer la opción Deshacer
- Reconocer la opción Rehacer
- Reconocer el lienzo
- Reconocer la opción escalar
- Reconocer la opción rotar
- Reconocer la opción voltear
- Reconocer la opción limpiar
- Reconocer la barra de herramientas
- Reconocer las opciones de área
- Reconocer la opción de intercambiar colores
- Reconocer la paleta de colores
- Reconocer la opción de acercar

Evaluación (logros)

- Sin ayuda de referencias, el estudiante indica cuáles son y para qué sirven las diferentes opciones del editor de pinturas de Scratch.

Períodos de clase: 2
Actividades:
TEMA: Actividad Introductoria a Scratch http://www.eduteka.org/proyectos.php/5/2483

Utilizar el editor de pinturas

- Importar una imagen de un archivo
- Dibujar sobre el lienzo
- Deshacer o rehacer una acción en el lienzo
- Escalar el tamaño de una imagen
- Rotar una imagen en el sentido de las manecillas del reloj
- Voltear una imagen de forma vertical o horizontal
- Limpiar un trazo dibujado en el lienzo

- Dibujar figuras con la brocha, la línea, el rectángulo y la elipse de la barra de herramientas.
- Utilizar el bote de pintura para llenar de color una figura
- Seleccionar el color con la herramienta de gotero
- Duplicar una imagen con la opción de estampar
- Borrar una imagen del lienzo
- Insertar un texto en el lienzo
- Mover un texto o una imagen dentro del lienzo
- Cambiar los tamaños de las herramientas en la opción de área
- Cambiar de color las imágenes con la paleta de colores.
- Usar la lupa para acercar una imagen

Evaluación (logros)

- Crea o edita una imagen, utilizando las principales opciones del editor de pinturas

Períodos de clase: 2

Actividades:

TEMA: Animando un paisaje en Scratch http://www.eduteka.org/proyectos.php/5/2447

Crear y editar Objetos, Disfraces, Fondos; y editar Escenario

- Pintar un Objeto nuevo (botón)
- Pintar, Importar, Editar, Copiar y Borrar Disfraces y Fondos.
- Mostrar al frente un Objeto < en lista de objetos >
- Exportar un Objeto

Evaluación (logros)

- Explica la diferencia en la ruta de acceso al editor de pinturas cuando se va a: crear un objeto, a editar uno de sus disfraces o a editar un fondo de un escenario.
- Crea nuevos objetos y edita sus disfraces.
- Realiza modificaciones sobre un fondo existente.

Períodos de clase: 2

Actividades:

TEMA: Animando un paisaje en Scratch http://www.eduteka.org/proyectos.php/5/2447

Dar instrucciones básicas a Objetos (al presionar, por siempre, esperar, mover, etc) – actividad 1

- Utilizar la instrucción al presionar Bandera Verde [Bloque Control]
- Utilizar la instrucción Por Siempre [Bloque Control]
- Utilizar la instrucción Esperar N segundos [Bloque Control]
- Utilizar las instrucciones Si y Si – Sino [Bloque Control]
- Explicar la instrucción Esperar Hasta Que [Bloque Control]
- Explicar la instrucción Por Siempre Si [Bloque Control]
- Utilizar la instrucción Mover N Pasos [Bloque Movimiento]
- Utilizar la instrucción Ir a X: Y: [Bloque Movimiento]
- Utilizar la instrucción Ir a <posición de un objeto> [Bloque Movimiento]
- Utilizar la instrucción Apuntar en dirección [Bloque Movimiento]
- Utilizar la instrucción Apuntar hacia <nombre objeto> [Bloque Movimiento]
- Utilizar la instrucción Rebotar si está tocando borde [Bloque Movimiento]
- Explicar la instrucción Girar N grados [Bloque Movimiento]
- Utilizar la instrucción Cambiar X por [Bloque Movimiento]
- Utilizar la instrucción Cambiar Y por [Bloque Movimiento]
- Utilizar la instrucción Fijar posición X [Bloque Movimiento]
- Utilizar la instrucción Fijar posición Y [Bloque Movimiento]
- Mostrar en el escenario la “posición X” de un objeto [Bloque Movimiento]
- Mostrar en el escenario la “posición Y” de un objeto [Bloque Movimiento]
- Mostrar en el escenario la “dirección” de un objeto [Bloque Movimiento]
- Utilizar la instrucción Deslizar en N segundos a posición X,Y [Bloque Movimiento]
- Utilizar el sensor “tocando <borde>” [Bloque Sensores]
- Utilizar la instrucción Cambiar efecto [Bloque Apariencia]
- Utilizar la instrucción Siguiente Disfraz [Bloque Apariencia]
- Utilizar la instrucción Fijar Tamaño a [Bloque Apariencia]
- Utilizar la instrucción Cambiar Tamaño por [Bloque Apariencia]
- Mostrar en el escenario el “tamaño” de un objeto [Bloque Apariencia]
- Mostrar en el escenario el “número de disfraz” de un objeto [Bloque Apariencia]
- Enviar un objeto hacia atrás N capas [Bloque Apariencia]
- Copiar el programa de un Objeto a otro
- Cambiar el nombre a un Objeto, Disfraz o Fondo
- Cambiar el nombre al Escenario
- Reconocer la posición de un Objeto
- Activar la opción “ver los pasos separados” en el botón EXTRAS
- Utilizar la ayuda en línea de Scratch
- Borrar instrucciones

Evaluación (logros)

- Utiliza, como mínimo, dos formas de mover un Objeto
- Al mover un objeto, incorpora al menos una instrucción repetitiva
- Incorpora instrucciones de Apariencia para enriquecer los movimientos de un Objeto.

Períodos de clase: 4

Actividades:

TEMA: Animando un paisaje en Scratch http://www.eduteka.org/proyectos.php/5/2447

Crear historias interactivas con Scratch incorporando instrucciones como: pensar, pensar por N segundos, decir, decir por N segundos, cambiar disfraz e instrucciones de sonido.

- Utilizar la instrucción Repetir [Bloque Control]
- Utilizar la instrucción Pensar [Bloque Apariencia]
- Utilizar la instrucción Pensar por N segundos [Bloque Apariencia]
- Utilizar la instrucción Decir [Bloque Apariencia]
- Utilizar la instrucción Decir por N segundos [Bloque Apariencia]
- Utilizar la instrucción Cambiar el Disfraz a [Bloque Apariencia]
- Utilizar la instrucción Cambiar el Fondo a [Bloque Apariencia]
- Utilizar la instrucción Fondo Siguiente [Bloque Apariencia]
- Utilizar la instrucción Enviar al Frente [Bloque Apariencia]
- Utilizar la instrucción Tocar sonido [Bloque Sonido]
- Utilizar la instrucción Tocar sonido y esperar [Bloque Sonido]
- Utilizar la instrucción Fijar Volumen a [Bloque Sonido]
- Utilizar la instrucción Cambiar Volumen por [Bloque Sonido]
- Utilizar la instrucción Tocar Tambor Durante N Pulsos [Bloque Sonido]
- Utilizar la instrucción Detener todos los sonidos [Bloque Sonido]
- Utilizar la instrucción Silenciar sonidos durante N pulsos [Bloque Sonido]
- Utilizar la instrucción Tocar nota <número nota> durante N pulsos [B. Sonido]
- Utilizar la instrucción Fijar instrumento a <instrumento> [Bloque Sonido]
- Mostrar en el escenario el Volumen [Bloque Sonido]
- Utilizar la instrucción Cambiar tempo por <valor> [Bloque Sonido]
- Utilizar la instrucción Fijar tempo a N pulsos por minuto [Bloque Sonido]
- Mostrar en el escenario el valor de Tempo [Bloque Sonido]
- Importar un fondo (Escenario)
- Importar un Objeto
- Importar un Disfraz
- Grabar, Importar, Reproducir, Borrar y Parar sonidos para un Objeto o Escenario.

Evaluación (logros)

- A partir de un diálogo entre dos personajes, redactado previamente, crea una animación en la que se reproduzca de manera sincronizada la interacción entre dichos personajes (movimientos y conversación).
- Incorpora sonidos a una animación existente, manejando tanto sonidos de fondo, como independientes para cada Objeto (personaje).

Períodos de clase: 5

Actividades:

TEMA: Diálogos en Scratch http://www.eduteka.org/proyectos.php/5/2448

Establecer diferencias entre eventos e hilos

- Explicar qué es un evento
- Entender en qué casos se requiere programar por eventos
- Comprender qué es un hilo
- Entender en qué casos se requiere uno o más hilos asociados a un Objeto

Evaluación (logros)

- Dado un proyecto de Scratch por el docente, señala partes de los programas que corresponden a "hilos"
- Explica en sus propias palabras qué es un hilo en Scratch
- Con base en un proyecto creado previamente, explica qué instrucciones corresponden a eventos y por qué son eventos.
- Explica en sus propias palabras cómo se puede utilizar un evento en un programa.

Períodos de clase: 2

Actividades:

TEMA: Interacción de Objetos en Scratch - Carrera carros http://www.eduteka.org/proyectos.php/5/2449

Crear programas que manejen eventos (sensores) y multihilos (enviar a todos, al recibir, al presionar objeto)

- Utilizar la instrucción Enviar a todos [Bloque Control]
- Utilizar la instrucción Enviar a todos [Bloque Control]
- Utilizar la instrucción Al Recibir [Bloque Control]
- Utilizar instrucción Al Presionar Objeto [Bloque Control]
- Utilizar la instrucción Al presionar tecla [Bloque Control]
- Utilizar la instrucción Repetir hasta que [Bloque Control]
- Utilizar la instrucción Detener Todo [Bloque Control]
- Utilizar la instrucción Detener Programa [Bloque Control]
- Utilizar la instrucción Mostrar [Bloque Apariencia]
- Utilizar la instrucción Esconder [Bloque Apariencia]
- Utilizar la instrucción Fijar Efecto a [Bloque Apariencia]
- Utilizar la instrucción Quitar Efectos Gráficos [Bloque Apariencia]
- Utilizar la instrucción Posición X del ratón [Bloque Sensores]
- Utilizar la instrucción Posición Y del ratón [Bloque Sensores]
- Utilizar la instrucción Ratón presionado? [Bloque Sensores]
- Utilizar el sensor <Tecla> Presionada [Bloque Sensores]
- Utilizar el sensor Tocando <...> [Bloque Sensores]
- Utilizar el sensor Tocando el Color <...> [Bloque Sensores]
- Utilizar la instrucción <color 1> sobre <color 2> [Bloque Sensores]
- Utilizar la instrucción Distancia a [Bloque Sensores]
- Utilizar la instrucción Reiniciar cronómetro [Bloque Sensores]
- mostrar Cronómetro en el escenario [Bloque Sensores]

- mostrar Volumen del sonido en el escenario [Bloque Sensores]
- mostrar resultado de ¿sonido fuerte?, en el escenario [Bloque Sensores]
- mostrar valor de un sensor, en el escenario [Bloque Sensores]
- mostrar en el escenario si un sensor está activado [Bloque Sensores]

Evaluación (logros)

- Dada una situación problema por el docente, desarrolla un programa en Scratch que incorpore al menos el manejo de dos eventos.

Períodos de clase: 8

Actividades:

TEMA: Interacción de Objetos en Scratch - Carrera carros http://www.eduteka.org/proyectos.php/5/2449

TEMA: Ampliando el juego Pong (parte 1) http://www.eduteka.org/proyectos.php/5/2458

Reconocer las formas de documentar la funcionalidad de un Proyecto en Scratch.

- Agregar comentarios a un programa
- Agregar notas al Proyecto

Evaluación (logros)

- Dado un proyecto elaborado previamente, le agrega comentarios a un Programa y notas al Proyecto.
- Explica brevemente la importancia de agregar comentarios a un Programa y notas a un Proyecto.

Períodos de clase: 1

Actividades:

TEMA: Ampliando el juego Pong (parte 1) http://www.eduteka.org/proyectos.php/5/2458

Realizar modificaciones a programas existentes para mejorarlos.

- Explicar qué hace cada una de las instrucciones de un programa.
- Agregar comentarios explicativos de la funcionalidad de un hilo (concepto general en lugar de instrucción por instrucción)
- Agregar o cambiar instrucciones que mejoren la funcionalidad o el desempeño del programa.

Evaluación (logros)

- Agrega comentarios explicativos de la funcionalidad de un Programa (hilo)
- Dado un proyecto en Scratch por el docente, y unos requerimientos de modificación sobre el mismo, agrega o cambia las instrucciones necesarias de tal manera que el programa cumpla con las nuevas especificaciones.

Períodos de clase: 1

Actividades:

TEMA: Ampliando el juego Pong (parte 1) http://www.eduteka.org/proyectos.php/5/2458

Elaborar dibujos mediante el movimiento de objetos (funcionalidades de Lápiz)

- Utilizar las instrucciones Subir Lápiz y Bajar Lápiz [Bloque Lápiz]
- Utilizar la instrucción Fijar Tamaño de Lápiz a [Bloque Lápiz]
- Utilizar la instrucción Fijar Color de Lápiz a [Bloque Lápiz]
- Utilizar la instrucción Fijar Intensidad de Lápiz a [Bloque Lápiz]
- Utilizar la instrucción Cambiar Tamaño de Lápiz por [Bloque Lápiz]
- Utilizar la instrucción Cambiar Color de Lápiz por [Bloque Lápiz]
- Utilizar la instrucción Cambiar Intensidad de Lápiz por [Bloque Lápiz]
- Utilizar la instrucción Borrar [Bloque Lápiz]
- Utilizar la instrucción Sellar (imagen del Objeto) [Bloque Lápiz]

Evaluación (logros)

- Utiliza la funcionalidad de lápiz para hacer dibujos sobre el escenario.

Períodos de clase: 4

Actividades:

TEMA: Ampliando el juego Pong (parte 2) http://www.eduteka.org/proyectos.php/5/2459

Utilizar operaciones matemáticas y booleanas

- Utilizar las operaciones " +, -, * , / " [Bloque Números]
- Utilizar la función Módulo [Bloque Números]
- Utilizar la función Redondear [Bloque Números]
- Utilizar funciones matemáticas (abs, raíz cuadrada, sin, cos, etc) [Bloque Números]
- Utilizar operaciones booleanas " >, <, =, y ,o ,no " [Bloque Números]
- Utilizar la instrucción Número al Azar entre 1 y N [Bloque Números]

Evaluación (logros)

- Utiliza las operaciones matemáticas y booleanas para resolver problemas matemáticos planteados por el docente.

Períodos de clase: 4

Actividades:

Crear y utilizar variables y listas

- Crear una Nueva Variable [Bloque Variables]
- Borrar una variable Bloque Variables]
- Utilizar la instrucción Fijar <variable> a <valor> [Bloque Variables]
- Utilizar la instrucción Cambiar <variable> por <valor> [Bloque Variables]
- Mostrar <variable> en el escenario [Bloque Variables]
- Utilizar la instrucción Esconder <variable> del escenario [Bloque Variables]
- Crear una Nueva Lista [Bloque Variables]
- Borrar una lista [Bloque Variables
- Mostrar <LISTA> en el escenario [Bloque Variables]

- Utilizar la instrucción Añade X a LISTA [Bloque Variables]
- Utilizar la instrucción Borrar (posición N) de LISTA [Bloque Variables]
- Utilizar la instrucción Insertar ELEMENTO en POSICIÓN N de LISTA [Bloque Variables]
- Utilizar la instrucción Reemplazar (posición N) de LISTA con NUEVO ELEMENTO [Bloque Variables]
- Utilizar la instrucción Item N de LISTA [Bloque Variables]
- Utilizar la instrucción Longitud de LISTA [Bloque Variables]

Evaluación (logros)

- Utiliza variables para apoyar la solución de problemas
- El estudiante crea y manipula listas para el manejo de elementos de información.

Períodos de clase: 6

Actividades:

Compartir con otras personas los trabajos realizados en Scratch.

- Explicar la importancia de compartir con otras personas los trabajos realizados en Scratch.
- Utilizar apropiadamente la instrucción Compartir

Evaluación (logros)

- Una vez creado un proyecto, lo comparte con otras personas a través de la página de Scratch.

Períodos de clase: 1

Actividades:

NOTA GENERAL: Componente curricular de Scratch elaborado por Eduteka con el apoyo de Motorola Foundation, Motorola de Colombia Ltda. y la gestión de la ONG Give to Colombia.

ANEXO 5

PROGRAMACIÓN DE COMPUTADORES
UNA PROPUESTA DE CURRÍCULO PARA MICROMUNDOS

Esta propuesta hace parte del "Modelo Curricular Interactivo de Informática"
http://www.eduteka.org/curriculo2/Herramientas.php?codMat=15

DEFINICIÓN
Algoritmos y programación se definen como los procesos de formulación de una solución a una situación planteada, apoyándose en conceptos y estructuras propias de la programación.

ALCANCE
Se busca que el estudiante utilice metodologías y estructuras secuenciales, iterativas y condicionales para analizar problemas, diseñar algoritmos, traducir algoritmos a un lenguaje de programación y depurar procedimientos sencillos con el fin de solucionar problemas. Los estudiantes no elaborarán programas complejos, sólo se concentrarán en la elaboración de procedimientos.

OBJETIVO GENERAL
Al terminar la instrucción en Algoritmos y Programación, el estudiante debe estar en capacidad de utilizar metodologías y estructuras secuenciales, iterativas y condicionales para analizar problemas, diseñar algoritmos, traducir algoritmos a un lenguaje de programación y depurar los procedimientos resultantes.

OBJETIVOS ESPECÍFICOS
Al finalizar la instrucción en esta herramienta informática, el estudiante estará en capacidad de:

≈ Comprender una metodología para resolver problemas matemáticos.

- Conocer los elementos que tienen en común la mayoría de los problemas matemáticos (estado inicial, meta, recursos y el estado actual de conocimientos de quien pretende resolverlos)
- Conocer las cuatro operaciones mentales que intervienen en la solución de problemas matemáticos (entender el problema, trazar un plan, ejecutarlo y revisar)
- Hacer conciencia sobre la utilización en la clase de matemáticas de estas cuatro operaciones para resolver problemas

PERÍODOS DE CLASE: 2
INDICADORES DE LOGRO: Sin ayuda de referencias, *describe* brevemente, y en sus propias palabras, las cuatro operaciones mentales que intervienen en la solución de problemas matemáticos. **[A, F]**
NOTA: Aunque Algoritmos y Programación se puede integrar con varias asignaturas, las matemáticas es un área muy adecuada ya que la forma de resolver problemas matemáticos se asemeja mucho al ciclo de programación.

≈ Comprender las etapas del ciclo de programación de computadores.

- Conocer las cuatro etapas del ciclo de programación para resolver problemas con ayuda del computador (analizar el problema, diseñar un algoritmo, traducir el algoritmo a un lenguaje de programación y depurar el programa)
- Comprender la similitud que hay entre las operaciones mentales que intervienen en la solución de problemas matemáticos y las etapas del ciclo de programación
- Entender que la solución de problemas matemáticos mediante programación tiene dos ciclos (uno en el que se resuelve el problema con lápiz y papel y otro en el que se automatiza la solución)
- Diferenciar entre Sistema Operativo y Software de Aplicación.
- Diferencias entre Software de Aplicación y Procedimientos.

PERÍODOS DE CLASE: 1
INDICADORES DE LOGRO: Explica brevemente las cuatro etapas del ciclo de programación para resolver problemas con ayuda del computador. **[A, F]**

≈ Comprender qué es un algoritmo.

- Comprender por qué no se debe empezar a diseñar un algoritmo hasta no haber analizado detalladamente los problemas que se desean resolver
- Identificar en el entorno: procesos, ciclos, rutinas o biorritmos que se puedan considerar como algoritmos (concepto intuitivo de algoritmo)
- Comprender que un algoritmo es un conjunto de pasos sucesivos y organizados en secuencia lógica
- Comprender la importancia de organizar en secuencia lógica los pasos de diversos procesos

PERÍODOS DE CLASE: 2
INDICADORES DE LOGRO: Describe al menos dos procesos, ciclos, rutinas o biorritmos que se den en el entorno y que puedan considerarse como algoritmos. **[A, F]**
Dada una lectura que describa una serie de instrucciones, sigue cada uno de los pasos indicados en esta, en el orden establecido. **[F]**

≈ Utilizar el lenguaje pseudocódigo para representar algoritmos.

- Utilizar frases o proposiciones en español para representar instrucciones
- Organizar en secuencia lógica las instrucciones que solucionan problemas planteados
- Refinar los algoritmos representados en pseudocódigo (escribir una primera versión y luego descomponerla en subproblemas, si fuera necesario)
- Comprender la importancia de detallar al máximo las instrucción para que estas se puedan traducir a un lenguaje de programación

PERÍODOS DE CLASE: 3

INDICADORES DE LOGRO: Dado un problema de la vida cotidiana (como hacer un jugo de fruta), construye un algoritmo en pseudocódigo para solucionarlo. **[A, F]**

≈ Comprender qué son identificadores, variables y constantes.

- Comprender que los identificadores son nombres que se dan a los elementos (variables, constantes, procedimientos) utilizados en los algoritmos
- Conocer un conjunto de reglas (convenciones) para asignar nombres a variables, constantes y procedimientos
- Conocer qué es una variable
- Entender cómo ayuda el uso de variables en la formulación de un algoritmo y en su utilización con diferentes conjuntos de datos iniciales (generalización)
- Conocer los tipos de variables y sus diferencias (globales y locales)
- Conocer cómo asignar un valor a una variable
- Conocer cómo utilizar el valor almacenado en una variable
- Conocer qué es una constante
- Conocer cómo asignar un valor a una constante
- Conocer cómo utilizar el valor almacenado en una constante

PERÍODOS DE CLASE: 6

INDICADORES DE LOGRO: Dado un problema de la vida cotidiana, lista las variables y constantes presentes en este. **[A, F]**
Dada una lista de variables y constantes, les asigna nombres que pueda entender el lenguaje de programación. **[A, F]**
Dada una lista de variables y constantes, indica cómo asignarles valores a estos. **[A, F]**

≈ Conocer los símbolos que se utilizan para representar algoritmos mediante diagramas de flujo.

- Comprender que los diagramas de flujo han sido una de las técnicas más utilizadas para representar gráficamente la secuencia de instrucciones de un algoritmo
- Identificar y recordar el significado de los principales símbolos estandarizados para elaborar diagramas de flujo (inicio, final, líneas de flujo, entrada por teclado, llamada a subrutina, saluda impresa, salida en pantalla, conector, decisión, iteración, etc)
- Conocer las principales reglas para elaborar diagramas de flujo (encabezado, dirección de flujo, iniciación de variables y constantes, etc)

PERÍODOS DE CLASE: 2

INDICADORES DE LOGRO: Dada una serie de símbolos para representar algoritmos, escribe al frente su significado. **[A, F]**
Dado un algoritmo sencillo, explica la función que realiza en cada uno de los pasos. **[A, F]**

≈ Reconocer el entorno de trabajo que ofrece un lenguaje de programación.

- Entender la barra de título
- Entender la barra de menús (Archivo, Edición, Ver, Insertar, Formato, Ventana)
- Entender las barras de herramientas
- Entender la barra de desplazamiento
- Entender la barra de estado
- Entender el área de trabajo

PERÍODOS DE CLASE: 1

INDICADORES DE LOGRO: En sus propias palabras, describe brevemente, el entorno de trabajo que ofrece el entorno de programación. **[A, F]**

≈ Utilizar apropiadamente las funciones básicas de un lenguaje de programación.

- Abrir y cerrar el ambiente de programación
- Abrir y cerrar un procedimiento existente
- Crear instrucciones nuevas dentro de un procedimiento existente
- Guardar un procedimiento en una unidad de almacenamiento local o remota
- Crear un proyecto nuevo
- Escribir, con la sintaxis correcta, instrucciones en el lenguaje de programación utilizado
- Compilar un procedimiento
- Ejecutar un procedimiento
- Utilizar las funciones de ayuda que ofrece el software

PERÍODOS DE CLASE: 3

INDICADORES DE LOGRO: Crea un nuevo procedimiento, lo graba en un lugar establecido por el profesor, lo cierra; si es necesario, lo abre nuevamente para modificarlo. **[A, F]**

ACTIVIDADES: Tema: Caricaturas http://eduteka.org/actividades/actividades.php?idH=501

≈ Traducir algoritmos a un lenguaje de programación.

- Identificar procedimientos que se utilicen frecuentemente en la vida diaria
- Conocer qué significa un procedimiento en un programa de computador
- Conocer la forma de elaborar un procedimiento con el lenguaje de programación seleccionado
- Comprender la estructura de un procedimiento (línea de título, instrucciones y final)
- Utilizar las reglas establecidas (convenciones) para nombrar procedimientos (identificadores)
- Conocer qué significa "palabra reservada"
- Conocer las principales primitivas (comandos) que ofrece el lenguaje de programación utilizado y tenerlas en cuenta para traducir los algoritmos a dicho lenguaje
- Conocer la sintaxis de las principales primitivas
- Conocer la forma de ejecutar un procedimiento en forma directa

- Conocer la forma de llamar un procedimiento desde otro procedimiento
- Hacer comentarios en procedimientos
- Traducir una a una las instrucciones de los diagramas de flujo al lenguaje de programación utilizado
- Elaborar procedimientos que acepten parámetros

PERÍODOS DE CLASE: 3

INDICADORES DE LOGRO: A partir de un algoritmo construido sobre un problema matemático, lo traduce a un procedimiento en MMP. **[A, F]**
Abre un procedimiento y lo ejecuta. **[A]**

ACTIVIDADES: Tema: Caricaturas http://eduteka.org/actividades/actividades.php?idH=501

≈ Utilizar el recurso de interactividad con los usuarios de los procedimientos.

- Entender qué es ser usuario de un programa de computador
- Comprender la importancia de la interactividad con el usuario en la generalización de soluciones a problemas
- Reconocer diferentes métodos de interacción con el usuario (teclado y ratón)
- Utilizar los comandos apropiados para establecer interactividad con el usuario mediante el teclado

PERÍODOS DE CLASE: 2

INDICADORES DE LOGRO: Elabora un procedimiento que solucione un problema planteado por el docente, en el que solicite al usuario digitar alguna información. **[F]**

≈ Reconocer los diferentes tipos de fallas que puede presentar un procedimiento.

- Comprender que hay fallas que detecta el computador (compilador) y otras no (fallas humanas)
- Conocer qué son las fallas de sintaxis y de precaución (detectables por el computador)
- Conocer qué son las fallas de concepción, de lógica y de procedimiento (fallas humanas)

PERÍODOS DE CLASE: 2

INDICADORES DE LOGRO: En sus propias palabras, describe brevemente, los tipos de fallas que se pueden presentar en un procedimiento. **[F]**

≈ Comprender en qué fases del ciclo de programación se pueden producir las fallas de sintaxis y de precaución.

- Comprender qué es una falla de sintaxis
- Entender en qué fase del ciclo de programación se pueden producir fallas de sintaxis
- Conocer las causas más comunes por las que se producen las fallas de sintaxis
- Comprender el significado de los mensajes de error que presenta el compilador cuando detecta una falla de sintaxis y solucionarla.
- Conocer las medidas que se deben tomar para evitar las fallas de sintaxis
- Comprender qué es una falla de precaución (recomendaciones técnicas o "warning error")
- Entender en qué fase del ciclo de programación se pueden producir fallas de precaución
- Conocer las causas más comunes por las que se producen las fallas de precaución
- Conocer las medidas que se deben tomar para evitar las fallas de precaución

PERÍODOS DE CLASE: 2

INDICADORES DE LOGRO: Dada una serie de fallas que se pueden presentar en un procedimiento, las relaciona con las fases del ciclo de programación en las cuales estas se pueden producir. **[F]**

≈ Comprender los pasos para analizar problemas.

- Conocer los pasos para analizar un problema que se quiere sistematizar mediante un procedimiento
- Comprender en qué consiste el paso "formular el problema" (determinar y comprender exactamente en qué consiste el problema)
- Comprender en qué consiste el paso "precisar los resultados esperados" (metas y submetas)
- Comprender en qué consiste el paso "identificar los datos disponibles"
- Comprender en qué consiste el paso "determinar las restricciones" (aquello que está permitido o prohibido hacer y/o utilizar para llegar a una solución)
- Comprender en qué consiste el paso "establecer los procesos necesarios" (operaciones)
- Hacer conciencia de cómo estos pasos ayudan a lograr el objetivo de la primera etapa del ciclo de programación

PERÍODOS DE CLASE: 2

INDICADORES DE LOGRO: Sin ayuda de referencias, lista los pasos propuestos para analizar problemas (formular el problema, precisar los resultados esperados, identificar los datos disponibles, determinar las restricciones y establecer los procesos necesarios) y describe brevemente en qué consiste cada uno. **[F]**

≈ Definir y utilizar variables y constantes en los algoritmos.

- Definir las variables y constantes necesarias para resolver un problema
- Nombrar las variables y constantes definidas utilizando las reglas establecidas para ello
- Inicializar las variables y constantes con los valores iniciales establecidos en el análisis del problema

PERÍODOS DE CLASE: 3

INDICADORES DE LOGRO: Dado un problema de la vida cotidiana, lista las variables y constantes presentes en este. **[A, F]**
Dada una lista de variables y constantes, les asigna nombres que pueda entender el lenguaje de programación. **[A, F]**
Dada una lista de variables y constantes, indica cómo asignarles valores a estos. **[A, F]**

≈ Comprender qué son operadores y expresiones.

- Conocer que es un operador
- Entender la clasificación de operadores (aritméticos, alfanuméricos, relacionales y lógicos).
- Saber el orden de evaluación de los operadores
- Conocer qué es una expresión
- Entender los elementos que pueden conformar una expresión (valores, funciones, primitivas (comandos), constantes, variables, cadenas alfanuméricas, operadores)
- Conocer diferentes tipos de expresiones (aritméticas, alfanuméricas, lógicas y de asignación)
- Comprender cómo se pueden unir varios de estos elementos mediante operadores para formar una expresión compuesta

PERÍODOS DE CLASE: 4

INDICADORES DE LOGRO: Dado un algoritmo por el docente, identifica y explica las expresiones y operadores presentes en este **[F]**

≈ Elaborar diagramas de flujo para representar soluciones de problemas.
- Utilizar símbolos para representar instrucciones
- Recordar y utilizar los principales símbolos estandarizados para elaborar diagramas de flujo (inicio, final, líneas de flujo, entrada por teclado, llamada a subrutina, salida impresa, salida en pantalla, conector, decisión, iteración, etc)
- Aplicar las reglas para elaborar diagramas de flujo (encabezado, dirección de flujo, iniciación de variables y constantes, etc)
- Organizar en secuencia lógica las instrucciones que solucionan problemas planteados
- Elaborar diagramas de flujo para representar soluciones de problemas
- Refinar los algoritmos mediante la escritura de una primera versión y luego descomponerla en subproblemas (procedimientos), si fuera necesario
- Detallar al máximo las instrucción para que estas se puedan traducir a un lenguaje de programación

PERÍODOS DE CLASE: 4

INDICADORES DE LOGRO: Dado un problema matemático (como sumar los números pares comprendidos entre 2 y 1.000), construye un algoritmo en forma de diagrama de flujo para solucionarlo. **[F]**

≈ Reconocer el entorno de trabajo que ofrece un software para elaborar diagramas de flujo (menús, barras, área de trabajo).
- Entender la barra de título
- Entender la barra de menús (Archivo, Edición, Ver, Insertar, Formato, Ventana)
- Entender las barras de herramientas
- Entender la barra de desplazamiento
- Entender la barra de estado
- Entender el área de trabajo
- Entender las opciones de zoom (aumentar/disminuir la escala de visualización)

PERÍODOS DE CLASE: 1

INDICADORES DE LOGRO: En sus propias palabras, describe brevemente, el entorno de trabajo que ofrece el software seleccionado para elaborar diagramas de flujo. **[A, F]**

NOTA: Ver la reseña de algunas herramientas descargables de Internet que facilitan el Aprendizaje Visual. Incluye descripción de software para construir Diagramas de Flujo http://www.eduteka.org/HerramientasVisuales.php

≈ Utilizar apropiadamente las funciones básicas de un software para elaborar diagramas de flujo.
- Abrir y cerrar la aplicación
- Abrir y cerrar un diagrama de flujo existente
- Crear instrucciones nuevas dentro de un diagrama de flujo utilizando el símbolo apropiado
- Crear líneas de flujo entre las instrucciones de un diagrama de flujo
- Adicionar un título general que identifique un diagrama de flujo
- Seleccionar título, instrucciones o líneas de flujo
- Mover de posición el título, las instrucciones o las líneas de flujo
- Eliminar título, instrucciones o líneas de flujo
- Utilizar el comando deshacer
- Guardar un diagrama de flujo en una unidad de almacenamiento local o remota
- Guardar un diagrama de flujo para que pueda abrirse con otras versiones del mismo software.
- Exportar un diagrama de flujo a un formato gráfico para que lo puedan leer otros programas
- Utilizar las funciones de ayuda que ofrece el software

PERÍODOS DE CLASE: 3

INDICADORES DE LOGRO: Utilizando un software para elaborar diagramas de flujo, crea un nuevo diagrama, lo graba en un lugar establecido por el profesor, lo cierra; si es necesario, lo abre nuevamente para modificarlo. **[A, F]**

≈ Realizar operaciones básicas con instrucciones y líneas de flujo.
- Editar el texto de instrucciones de un diagrama de flujo
- Cambiar la apariencia de las instrucciones (color, fuente, tamaño, forma (símbolo), etc)
- Cambiar la apariencia de las líneas de flujo (color, grosor de la línea, aspecto de la flecha, etc)
- Utilizar las opciones de copiar y pegar para duplicar instrucciones
- Utilizar la opción que ofrece el software para organizar automáticamente los diagramas de flujo

PERÍODOS DE CLASE: 1

INDICADORES DE LOGRO: Dado un problema por el profesor, elabora un procedimiento para solucionarlo; la solución debe incluir el análisis del problema y el diagrama de flujo. **[F]**

≈ Elaborar procedimientos con estructura secuencial.
- Conocer qué es una estructura secuencial
- Conocer qué tipo de instrucciones puede contener una estructura secuencial (declaración de variables y constantes, asignación de valores, entrada de datos, operaciones, reporte de resultados)
- Utilizar el orden correcto de ejecución de las instrucciones del algoritmo
- Elaborar procedimientos que contengan únicamente la estructura secuencial
- Reflexionar sobre la estructura utilizada en la solución de los problemas

PERÍODOS DE CLASE: 6

INDICADORES DE LOGRO: A partir de un algoritmo construido sobre un problema matemático, lo traduce a un procedimiento en MMP. **[A, F]**
Abre un procedimiento y lo ejecuta. **[A]**
Elabora un procedimiento que solucione un problema planteado por el docente, en el que solicite al usuario digitar alguna información. **[F]**
Sin ayuda de referencias, describe con sus propias palabras qué es en programación una estructura secuencial. **[F]**

≈ Identificar los mensajes de error más comunes que presenta el lenguaje de programación utilizado (fallas de sintaxis) y si los hay corregirlos.

- Comprender el significado de los mensajes de error que presenta el compilador cuando detecta una falla de sintaxis
- Realizar acciones correctivas en el programa cuando el compilador reporte una falla de sintaxis

PERÍODOS DE CLASE: 2
INDICADORES DE LOGRO: Dado por el docente un procedimiento que contenga fallas de sintaxis, identifica y corrige dichas fallas. **[F]**

≈ Comprender en qué fases del ciclo de programación se pueden producir las fallas de concepción y de lógica.

- Comprender qué es una falla de concepción (mala formulación del problema)
- Entender en qué fase del ciclo de programación se pueden producir fallas de concepción
- Conocer las causas más comunes por las que se producen las fallas de concepción
- Conocer las medidas que se deben tomar para evitar las fallas de concepción
- Comprender qué es una falla de lógica (algoritmos mal diseñados)
- Entender en qué fase del ciclo de programación se pueden producir fallas de lógica
- Conocer las causas más comunes por las que se producen las fallas de lógica
- Conocer las medidas que se deben tomar para evitar las fallas de lógica

PERÍODOS DE CLASE: 2
INDICADORES DE LOGRO: Dado un algoritmo elaborado por otro estudiante, realiza la prueba de escritorio y predice que resultado arrojará el computador con un conjunto determinado de datos de entrada. **[F]**

≈ Realizar prueba de escritorio a los algoritmos elaborados.

- Comprender qué es una prueba de escritorio para un algoritmo
- Comprender cómo se realiza una prueba de escritorio
- Realizar la prueba de escritorio a los algoritmos diseñados (dando diferentes datos de entrada y siguiendo la secuencia indicada en el diagrama)

PERÍODOS DE CLASE: 2
INDICADORES DE LOGRO: Dado un algoritmo elaborado por otro estudiante, realiza la prueba de escritorio y predice que resultado arrojará el computador con un conjunto determinado de datos de entrada. **[F]**

≈ Analizar problemas utilizando una metodología con pasos ordenados.

- Listar en forma ordenada los pasos indicados para analizar problemas (“formular el problema”, “precisar los resultados esperados”, “identificar los datos disponibles”, “determinar las restricciones” y “establecer los procesos necesarios”).
- Comprender que los pasos de la metodología para analizar problemas son dinámicos y cíclicos (no es necesario seguirlos en forma secuencial)
- Comprender la importancia y conveniencia de emplear una metodología para analizar problemas

PERÍODOS DE CLASE: 3
INDICADORES DE LOGRO: Dado un problema por el docente, realiza el primer paso de análisis (formular problemas) siguiendo las indicaciones suministradas en clase. **[F]**

≈ Realizar el primer paso de la etapa de análisis: “Formular problemas”.

- Formular por escrito problemas a partir de situaciones de la vida real, planteadas en forma verbal
- Hacer conciencia sobre la naturaleza ambigua, imprecisa, incompleta e incongruente que en muchas ocasiones tiene el lenguaje natural y cómo afecta esto la formulación de un problema
- Determinar si se puede definir mejor los problemas planteados
- Identificar y buscar en el diccionario las palabras desconocidas que aparecen en los problemas
- Reflexionar sobre si se ha resuelto problemas similares con anterioridad

PERÍODOS DE CLASE: 3
INDICADORES DE LOGRO: Dado un problema por el docente, realiza el primer paso de análisis (formular problemas) siguiendo las indicaciones suministradas en clase. **[F]**

≈ Realizar el segundo paso de la etapa de análisis: “Precisar el resultado esperado”.

- Precisar con claridad cuál es resultado final (producto) que debe devolver el programa elaborado
- Establecer el formato que debe tener el resultado final (impreso, en pantalla, diagramación, orden, etc)
- Identificar la información relevante de un problema

PERÍODOS DE CLASE: 3
INDICADORES DE LOGRO: Dado un problema por el docente, realiza el segundo paso de análisis (precisar el resultado esperado) siguiendo las indicaciones suministradas en clase. **[F]**

≈ Realizar el tercer paso de la etapa de análisis: “Identificar los datos disponibles”.

- Identificar cuál es la información importante, para llegar a una solución, que se ofrece en la formulación de problemas
- Identificar cuál es la información no relevante, para llegar a una solución, que se ofrece en la formulación de problemas (que se puede omitir)
- Identificar los datos de entrada (conocidos) y la(s) incógnita(s) (datos desconocidos)
- Establecer las categorías en las cuales se pueden agrupar los datos
- Determinar el nivel de conocimiento que se posee en el ámbito de los problemas que se pretende resolver y establecer una estrategia para obtener los conocimientos que no se tienen actualmente, necesarios para llegar a una solución

PERÍODOS DE CLASE: 3
INDICADORES DE LOGRO: Dado un problema por el docente, realiza el tercer paso de análisis (identificar los datos disponibles) siguiendo las indicaciones suministradas en clase. **[F]**

≈ Realizar el cuarto paso de la etapa de análisis: “Determinar las restricciones”.

- Determinar lo que está permitido o prohibido hacer y/o utilizar para llegar a una solución (restricciones)
- Identificar las condiciones que se plantean en la formulación de los problemas
- Identificar los datos que pueden considerarse como fijos (constantes)
- Identificar los datos que deben considerarse como variables
- Identificar los datos que deben calcularse

PERÍODOS DE CLASE: 3

INDICADORES DE LOGRO: Dado un problema por el docente, realiza el cuarto paso de análisis (determinar las restricciones) siguiendo las indicaciones suministradas en clase. **[F]**

≈ Realizar el quinto paso de la etapa de análisis: "Establecer los procesos necesarios".

- Determinar los procesos (operaciones) que permiten llegar a los resultados esperados a partir de los datos disponibles
- Determinar las fórmulas que deben emplearse
- Identificar como afectan las condiciones (restricciones) a los procesos
- Identificar el orden en el que deben realizarse las operaciones
- Dividir, si es el caso, un problema en otros más pequeños y fáciles de solucionar (procedimientos)

PERÍODOS DE CLASE: 3

INDICADORES DE LOGRO: Dado un problema por el docente, realiza el quinto paso de análisis (establecer los procesos necesarios) siguiendo las indicaciones suministradas en clase. **[F]**

≈ Utilizar operadores y construir expresiones.

- Utilizar operadores aritméticos para construir expresiones aritméticas que tengan en cuenta el orden de evaluación de los operadores
- Utilizar operadores alfanuméricos para construir expresiones alfanuméricas
- Utilizar operadores lógicos y relacionales para construir expresiones lógicas
- Utilizar combinaciones apropiadas de operadores para construir expresiones de asignación

PERÍODOS DE CLASE: 3

INDICADORES DE LOGRO: Dado un problema por el docente que requiera plantear expresiones, identifica variables y constantes y las une mediante operadores para establecer las expresiones correctas que resuelvan el problema. **[F]**

≈ Elaborar procedimientos con estructura iterativa.

- Conocer qué es una estructura iterativa
- Comprender en qué casos es ventajoso utilizar una estructura iterativa
- Conocer qué tipo de instrucciones puede contener una estructura iterativa (instrucciones de control de ciclo, todas las instrucciones de la estructura secuencial)
- Conocer los comandos con los cuales se implementa la estructura iterativa
- Utilizar el orden correcto de ejecución de las instrucciones del algoritmo
- Elaborar procedimientos con una estructura iterativa que contenga y controle a una estructura secuencial
- Reflexionar sobre los tipos de problemas que requieren utilizar la estructura iterativa en la solución

PERÍODOS DE CLASE: 6

INDICADORES DE LOGRO: Sin ayuda de referencias, describe con sus propias palabras qué es en programación una estructura iterativa (de repetición). **[F]**
Dado por el docente un problema que requiera para su solución una estructura iterativa (repetición), elabora un procedimiento con una estructura iterativa que contenga y controle una estructura secuencial; la solución debe incluir el análisis del problema, el diagrama de flujo y la prueba de escritorio. **[F]**

≈ Identificar fallas de lógica en los algoritmos elaborados y sí las hay corregirlas.

- Comprender las fallas de lógica no son detectables por el compilador
- Realizar acciones correctivas en el programa cuando mediante la prueba de escritorio se encuentre una falla de lógica

PERÍODOS DE CLASE: 2

INDICADORES DE LOGRO: Dado por el docente un algoritmo y el respectivo procedimiento los cuales contengan fallas de lógica, identifica y corrige dichas fallas. **[F]**

≈ Comprender la importancia de verificar los resultados que produce un programa de computador.

- Comprender la importancia de verificar resultados
- Comprender cómo se verifican los resultados de un programa

PERÍODOS DE CLASE: 2

INDICADORES DE LOGRO: Dado por el docente un algoritmo y el respectivo procedimiento los cuales contengan fallas de lógica, identifica y corrige dichas fallas. **[F]**
Sin ayuda de referencias, explica con sus propias palabras cuál es la importancia de verificar los resultados que produce un programa de computador. **[F]**

≈ Reflexionar sobre la conveniencia de emplear una metodología con pasos ordenados para analizar problemas.

- Hacer conciencia de la importancia y conveniencia de emplear una metodología para analizar problemas
- Reflexionar sobre la importancia de poner por escrito el resultado del análisis de problemas (para cada uno de los pasos)

PERÍODOS DE CLASE: 1

INDICADORES DE LOGRO: Elabora un ensayo en el que plasma sus reflexiones cobre la conveniencia o no de emplear una metodología con pasos ordenados para aprender a analizar problemas. **[F]**

≈ Comprender qué tipos de datos acepta el lenguaje de programación utilizado.

- Conocer diferentes tipos de datos (números, palabras, listas, arreglos, etc)
- Conocer qué tipos de datos acepta el lenguaje de programación utilizado y qué tratamiento le da a cada tipo

PERÍODOS DE CLASE: 3

INDICADORES DE LOGRO: Sin ayuda de referencias, contrasta las diferencias entre los tipos de datos que acepta el lenguaje de programación utilizado. **[F]**

≈ Elaborar procedimientos con estructura condicional.

- Conocer qué es una estructura condicional (selección simple y doble)
- Comprender en qué casos es ventajoso utilizar una estructura condicional
- Conocer qué tipo de instrucciones puede contener una estructura condicional (instrucciones de decisión, de control de ciclo y todas las instrucciones de la estructura secuencial)
- Conocer y utilizar correctamente los comandos con los cuales se implementa la estructura condicional de selección simple
- Conocer y utilizar correctamente los comandos con los cuales se implementa la estructura condicional de selección doble
- Comprender que las proposiciones utilizadas en la estructura condicional deben poder evaluarse como verdaderas o falsas (solo dos valores posibles y excluyentes)
- Utilizar correctamente los operadores relacionales y lógicos para construir proposiciones (sencillas y compuestas)
- Expresar apropiadamente las proposiciones para que el lenguaje de programación las pueda entender y evaluar
- Utilizar el orden correcto de ejecución de las instrucciones del algoritmo
- Elaborar procedimientos con una estructura condicional que contenga y controle a una estructura secuencial
- Reflexionar sobre el papel que cumple el lenguaje en la formulación y uso de relaciones de orden y de proposiciones
- Reflexionar sobre los tipos de problemas que requieren utilizar la estructura condicional en la solución
- Reflexionar sobre la importancia que tiene reconocer las estructuras de solución de problemas en la forma de planear secuencias de acciones

PERÍODOS DE CLASE: 6

INDICADORES DE LOGRO: Sin ayuda de referencias, describe con sus propias palabras qué es en programación una estructura condicional. **[F]**
Dado por el docente un problema que requiera para su solución una estructura condicional, elabora un procedimiento con una estructura condicional que contenga y controle una estructura secuencial; la solución debe incluir el análisis del problema, el algoritmo en forma de diagrama de flujo y la prueba de escritorio. **[F]**

≈ Verificar los resultados que produce un procedimiento.

- Estimar el resultado que debe producir un procedimiento y compararlo con el realmente producido para determinar si se encuentra cercano al valor estimado
- Realizar todas las operaciones manualmente con un conjunto de datos iniciales y comparar el resultado con el que arroja el procedimiento luego de introducirle el mismo conjunto de datos

PERÍODOS DE CLASE: 2

INDICADORES DE LOGRO: A partir de la lectura de problemas, identifica correctamente el resultado esperado, los datos disponibles, las restricciones y los procesos necesarios para resolverlos. **[F]**
Dada una situación del mundo real, enuncia (formula) un problema que tenga todos los elementos (resultado esperado, datos disponibles, restricciones y procesos necesarios) y explica por qué es un problema. **[F]**

≈ Comprender la importancia de documentar los procedimientos.

- Comprender qué es la documentación de procedimientos y las ventajas que ofrece
- Entender que los procedimientos documentados son más fáciles de leer y comprender por otras personas diferentes a quien los escribió

PERÍODOS DE CLASE: 1

INDICADORES DE LOGRO: Sin ayuda de referencias, explica brevemente y con sus propias palabras cuál es la importancia de documentar los procedimientos. **[F]**

≈ Comprender la importancia de ajustar (afinar o mejorar) los procedimientos.

- Comprender en qué consiste ajustar un procedimiento
- Comprender que los requerimientos iniciales pueden estar mal planteados, por tanto el procedimiento resultante se puede mejorar
- Conocer el impacto que tiene en la imagen de un programador la realización de mejoras en los procedimientos (ir más allá de lo solicitado)

PERÍODOS DE CLASE: 2

INDICADORES DE LOGRO: Sin ayuda de referencias, explica brevemente y con sus propias palabras cuál es la importancia de ajustar (afinar o mejorar) los procedimientos. **[F]**

NOTA GENERAL: El número de clases que aparece asociada a cada uno de los objetivos específicos indica la cantidad de períodos de clase, de 55 minutos, estimado por los profesores participantes, necesarios para impartir la instrucción y realizar ejercicios y prácticas necesarios para que el estudiante pueda dominar lo que se pretende enseñar.

ESTÁNDARES

A. *Operaciones y Conceptos Básicos*
B. *Problemas Sociales, Éticos y Humanos*
C. *Herramientas de las TIC para la Productividad*
D. *Herramientas de las TIC para la Comunicación*
E. *Herramientas de las TIC para la Investigación*
F. *Herramientas de las TIC para la Solución de Problemas y la Toma de Decisiones*

ANÁLISIS DEL PROBLEMA

Formular el problema :

Resultados esperados :

Datos Disponibles :

Restricciones :

Procesos Necesarios :

ANEXO 7

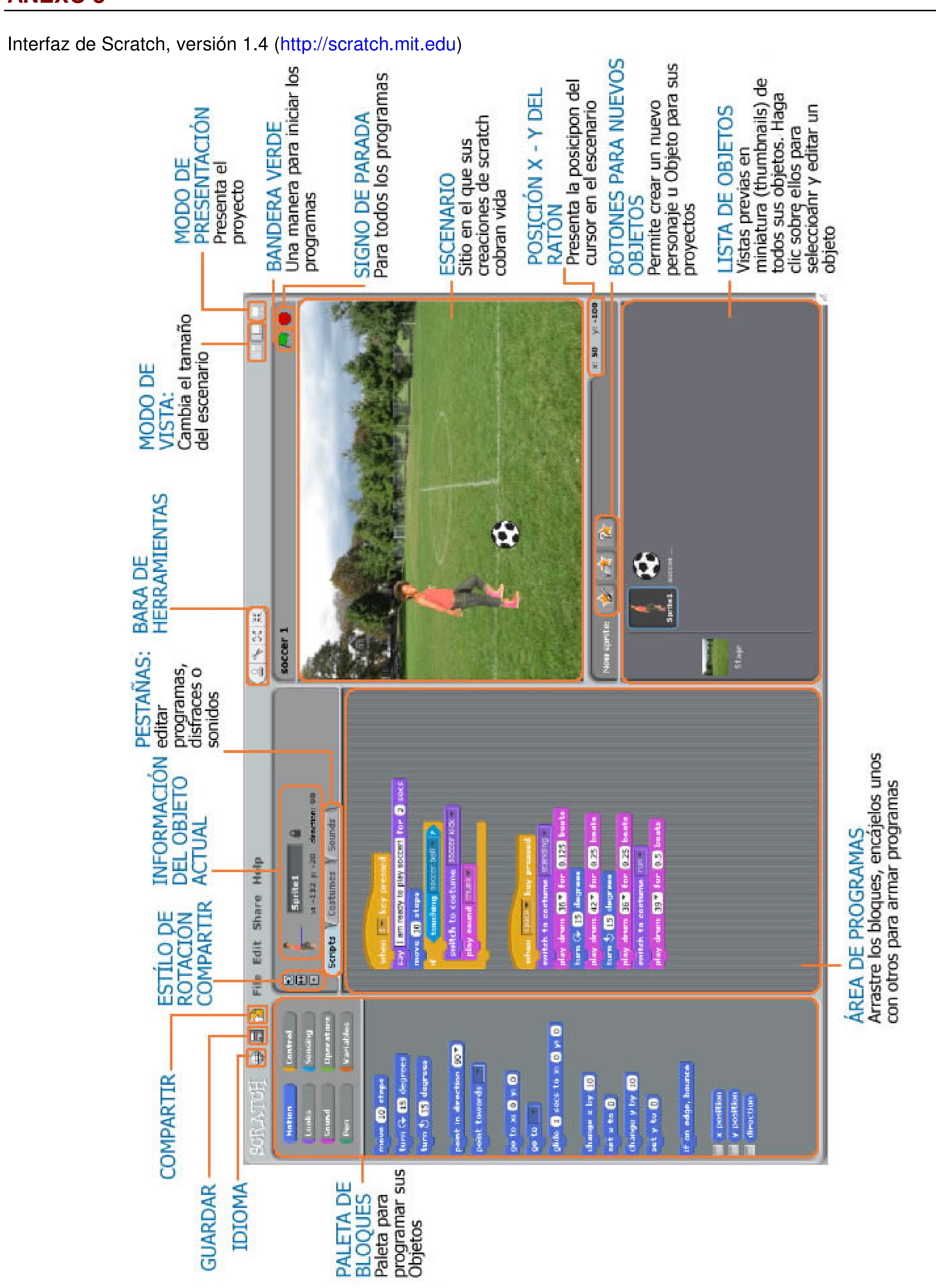
CÓDIGO EN MICROMUNDOS

DIAGRAMA DE FLUJO

SEUDOCÓDIGO

ANEXO 8

Interfaz de Scratch, versión 1.4 (http://scratch.mit.edu)



BIBLIOGRAFÍA CITADA O CONSULTADA

21st Century Skills (2004): ***Logros indispensables para los estudiantes del Siglo XXI***. [Consulta en línea: Eduteka, Marzo 16, 2009, http://www.eduteka.org/SeisElementos.php]

Aebli, Hans (2001): ***12 Formas básicas de enseñar, una didáctica basada en la psicología***; Ediciones Narcea, Madrid.

Banaji, Shakuntal & Burn, Andrew (2006): ***The rhetorics of creativity: a review of the literature***. Centre for the Study of Children, Youth and Media, Institute of Education (University of London), Londres [consulta en línea: Creative Partnership, Abril 13, 2009, http://www.creative-partnerships.com/data/files/rhetorics-of-creativity-12.pdf]

Beltrán, Luis Pompilio & Suárez, Alberto (1999): ***Matemáticas con tecnología aplicada 5***; Prentice Hall, Bogotá.

Bernays, P (1949): ***Lógica y ciencia***; Actas del congreso internacional de filosofía de la ciencia, Colloque de Logique, Paris.

Brandsford, John & Stein, Barry (1984): ***The IDEAL problem solver***, [Consulta en línea de un resumen del contenido del libro: npsnet, Marzo 27, 2004, http://www.npsnet.com/waletzky/BookSummaries/TheIDEALProblemSolver.html]

Brown, Stephen & Walter, Marion (1990): ***The art of problem posing***, [Consulta en línea: Questia, Marzo 27, 2007, http://www.questia.com/PM.qst?a=o&d=58818239]

Bunge, Mario (1959): ***Metascientific queries***; Ed. Charles C. Thomas, Springfield.

Bustamante Arias, Alfonso (2007): ***Notas para un curso de Lógica y argumentación***; Universidad Icesi, Cali, primera edición.

Cajaraville Pegito, José A. (1989): ***Ordenador y educación matemática, algunas modalidades de uso***; Editorial Síntesis, Madrid.

Cantillo Parra, Lucila (1990): ***Matemática concreta 5, cuaderno de actividades***; Editorial Voluntad, Bogotá.

Caro Pineda, Silvina (2003): ***Lógica de programación y algoritmos***; Centro de investigaciones para el desarrollo CIPADE, Uniboyacá, Tunja.

Carreras Llorenç y Otros (2001): ***Cómo educar en valores***; Narcea Ediciones, Madrid.

Casasbuenas, Cecilia & Eslava, Carola (1985): ***Cubo mágico 4***; Educar Editores, Bogotá.

Casasbuenas, Cecilia & Cifuentes, Virginia (1998a): ***Cuenta jugando 4***; Editorial Norma, Bogotá.

Casasbuenas, Cecilia & Cifuentes, Virginia (1998b): ***Cuenta jugando 5***; Editorial Norma, Bogotá.

Castellanos, Ricardo & Ferreira, Gonzalo (2000a): ***Informática 1***; Editorial Alfa Omega, Bogotá.

Castellanos, Ricardo & Ferreira, Gonzalo (2000b): ***Informática 2***; Editorial Alfa Omega, Bogotá.

Castellanos, Ricardo & Ferreira, Gonzalo (2000c): ***Informática 3***; Editorial Alfa Omega, Bogotá.

Castellanos, María Victoria de & Torres, Gladis (1986): ***Sistema matemático 4***; Editorial Norma, Bogotá.

Castiblanco Paiba, Ana Cecilia & Castiblanco Paiba, José Antonio (1988): ***Practiquemos matemática 5***; Intermedio Editores, Bogotá.

Clements, Douglas H. & Meredith, Julie S. (1992): ***Research on Logo, effects and efficacy***, [Consulta en Línea: Logo Foundation, MIT, Febrero 26 de 2007, http://el.media.mit.edu/logo-foundation/pubs/papers/research_logo.html]

Copi, Irving & Cohen, Carl (2000): ***Introducción a la Lógica***; Editorial Limusa, México.

Corbí Bellot, Antonio y Otros (1998): ***Fundamentos de programación, Volumen I: Metodología***; Universidad de Alicante, España.

Craft, Ana (2001): ***An Analysis of Research and Literature on Creativity in Education***. Qualifications and Curriculum Authority.

Creative Partnerships (2006): ***The rhetorics of creativity: a review of the literature***; Institute of Education, University of London; [Consulta en Línea: Creative Partnerships, Abril 13 de 2009, http://www.creative-partnerships.com/research-resources/]

Cuena, José (1986): ***Inteligencia artificial, sistemas expertos (sistemas basados en reglas y programación lógica);*** Alianza Editorial, Madrid.

Daintith, John (1982): ***Diccionario de matemáticas***; Editorial Norma, Bogotá.

De Bono, Edward (1970): ***El pensamiento lateral***, Editorial Paidos Ibérica, Barcelona.

---------- (1992): ***El pensamiento creativo***, Editorial Paidos Ibérica, Barcelona.

Del Rio Gómez, Sara Luz (2003): ***Técnica de solución de problemas utilizando una computadora***; [Consulta en Línea: Unam, Enero 15 de 2007, http://www.bibliodgsca.unam.mx/tesis/tes9sarg/toc.htm]

Delors, Jacques (1996): ***La educación encierra un tesoro***; Informe de la Unesco de la comisión internacional sobre la educación para el siglo XXI; Grupo Santillana, Madrid.

Deval, Juan (2001): ***Aprender en la vida y en la escuela***; Ediciones Morata, Madrid, segunda edición.

Díaz Pulecio, Laura Jeannette (1993): ***Recreo matemático 5, cuaderno de actividades***; Editorial Voluntad, Bogotá.

Edie, Arvid R. Y Otros (1997): ***Engineering fundamentals and problem solving***; McGraw Hill, Estados Unidos, tercera edición.

Feicht, Louis (2000): ***Old computer tricks: Enhance algebraic thinking***; Learning & Leading with technology, Volumen 27, Número 8.

Ferrater Mora, José (1957): ***¿Qué es la lógica?;*** Editorial Columba, Argentina.

Gallardo Ruiz, José & García López, Carmen: ***Apuntes para la asignatura informática, diseño de algoritmos y programas***; [Consulta en línea: Universidad de Málaga, Diciembre 8 de 2004, http://www.lcc.uma.es/personal/pepeg/mates]

Gallo, Gonzalo (2004): ***El sentido de la vida***; Periódicos Asociados, Bogotá.

Gardner, Howard (1993): ***Creating minds: An anatomy of creativity seen through the lives of Freud, Einstein, Picasso, Stravinsky, Eliot, Graham and Gandhi***. Basic Books, Nueva York.

Garza, Rosa María & Leventhal, Susana (2000): ***Aprender cómo aprender***; Editorial Trillas, México.

Good, Thomas & Brophy, Jere (1996): ***Psicología Edutactiva Contemporánea***; McGraw Hill, México, quinta edición.

Gutiérrez, José Blun & Montenegro, Ignacio (1995): ***Juguemos con Logo 2***; Fecón Ltda., Bogotá.

Guzdial, Mark (2000): ***Soporte tecnológico para el aprendizaje basado en proyectos***, Capítulo 3 del libro Aprendiendo con tecnología, Crhis Dede (compilador); Paidos, Argentina.

Holloway, G.E.T. (1982): ***Percepción del espacio en el niño según Piaget***; Ediciones Piados Ibérica, Barcelona.

Iranzo, Pascual Julián (2005): ***Lógica simbólica para informáticos***; Alfaomega, México.

ISTE (2007): ***Estándares Nacionales Estadounidenses de TIC para Estudiantes***. [Consulta en línea: Eduteka, Marzo 18, 2009, http://www.eduteka.org/estandaresestux.php3]
Jiménez Collazos, Luz Elena (2002): ***Conceptos básicos de programación con Java***; Universidad Icesi, Cali.
Johansson, Frans (2005): ***El efecto Medici***. Ediciones Deusto, Barcelona.
Jonassen, David; Carr, Chad; Yue, Hsiu-Ping (1998): ***Computadores como herramientas de la mente***; [Consulta en línea: Eduteka, Abril 7 de 2007, http://eduteka.org/tema_mes.php3?TemaID=0012]
Joyanes Aguilar, Luis (2001): ***Fundamentos de programación, algoritmos y estructura de datos***; Mc Graw Hill, México, segunda edición.
Lau, Joe & Chan, Jonathan (2004): ***OpenCourseWare on critical thinking, logic, and creativity: Strategic reasoning***; [Consulta en línea: Critical Thinking Web, Junio 15 de 2007, http://philosophy.hku.hk/think/strategy/]
Lizcano de Guerrero, Carmen (1999): ***Plan curricular***; Ediciones Universidad Santo Tomás, Bogotá, segunda edición.
López, Eliana (2004): ***La evaluación en educación en valores***; [Consulta en línea: Organización de Estados Iberoamericanos OEI, Agosto 31 de 2004, http://www.campus.oei.org/valores/boletin10a02.htm]
López Fríaz, Blanca Silvia (2000): ***Pensamientos crítico y creativo***. Trillas, México.
López García, Juan Carlos (2009): ***Programación de computadores y creatividad***; Guía de Algoritmos y Programación para docentes [Consulta en línea: Eduteka, Julio 16, 2009, http://www.eduteka.org/ProgramacionCreatividad.php].
Malan, David (2007): ***Scratch para los futuros científicos de la computación***; Ponencia presentada en el octavo simposio técnico de ACM, Covington, Kentucky [Consulta en línea: Eduteka, Noviembre 9, 2009, http://www.eduteka.org/ScratchMalan.php].
Marquínez Argote, Germán & Sanz Adrados, Juan José (1988): ***Lógica***; Universidad Santo Tomás, Bogota.
Melo R., Clara Esther (2001); ***Dominios 5, matemáticas para básica primaria***; Editorial Escuelas del Futuro, Bogotá.
MEN (2003): **Estándares básicos de calidad en matemáticas y lenguaje**, versión adaptada para las familias colombianas; Ministerio de Educación Nacional de Colombia (MEN); Proyecto MEN-ASCOFADE, Bogotá; [Consulta en línea: MEN, Febrero 16 de 2004, http://www.eduteka.org/pdfdir/MENEstMatLen.pdf]
---------- (2000): ***Lineamientos curriculares para el área de tecnología e informática***; Ministerio de Educación Nacional de Colombia (MEN).
---------- (1999): ***Nuevas tecnologías y currículo de matemáticas, lineamientos curriculares***; Cooperativa Editorial Magisterio, Bogotá.
---------- (1996): ***Educación en tecnología, propuesta para la educación básica***; Documento I; Ministerio de Educación Nacional de Colombia (MEN).
Moursund, David (1999): ***Project-Based learning using information technology***; ISTE Publications.
---------- (1996): ***Increasing your expertise as a problem solver, some roles of computers***; [Consulta en línea: Universidad de Oregon, Enero 28 de 2004, http://darkwing.uoregon.edu/~moursund/PSBook1996/introduction.htm]
---------- (2006): ***Computational Thinking and Math Maturity: Improving Math Education in K-8 Schools***; [Consulta en línea: Universidad de Oregon, Octubre 25 de 2007, http://uoregon.edu/~moursund/Books/ElMath/ElMath.html]
NAP (2004): ***Computer Science: Reflections on the field, reflections from the field***; [consulta en línea: National Academy Press; http://nap.edu/catalog/11106.html]
NRC -National Research Council- (2004): ***Being fluent with information technology***; National Academy Press (NAP), Washington, D.C. ; [Consulta en línea: NAP, Enero 30 de 2005, http://www.nap.edu/html/beingfluent/]
Niño, Carlos Alberto & Rodríguez, Beryeny (1999): ***Mi libro matemático 5***; Ediciones Magíster, Bogotá.
Olmos Gil, Tulio (2003): ***Lógica para niños***; [Consulta en línea: Aldea Educativa, Enero 08 de 2004, http://www.terra.com.ve/aldeaeducativa/temas/tareas28726.html]
Onrubia, Javier & Rochera, Maria José & Barberà, Elena (2001): ***La enseñanza y el aprendizaje de las matemáticas: una perspectiva psicológica***; capítulo 19 del libro Desarrollo psicológico y educación, César Coll (compilador), Alianza Editorial, Madrid.
Ortiz de Maschwitz, Elena María (2000): ***Inteligencias múltiples en la educación de la persona***; Editorial Magisterio, Bogotá.
Papert, Seymour (1993): ***Mindstorms: Children, computers, and powerful ideas***; Basic Books, New York, segunda edición.
Piaget, Jean (1964): ***Seis estudios de psicología***; Seix Barral, Barcelona, 1977 (Éditions Gonthier, Ginebra, 1964).
---------- (1969a): ***Biología y conocimiento***; Siglo XXI Editores, México.
---------- (1993): ***Estudios sobre lógica y psicología*** (compilación Alfredo Deaño y Juan Delval); Ediciones Altaya, Barcelona.
Piaget, Jean & Inhelder Bärbel (1969b): ***Psicología del niño***; Ediciones Morata, Madrid.
Polya, George (1957): ***How to solve it***; Princeton University Press, segunda edición.
Ramírez Bohórquez, Tania (2004): ***Deje atrás el temor a la matemática***; Editorial Nuevo Mundo, Bogotá; Tercera Edición.
Resnick, Mitchel (2007a): ***Sembrando las semillas para una sociedad más creativa***. Laboratorio de medios de MIT, Massachussets [Consulta en línea: Eduteka, Marzo 16, 2009, http://www.eduteka.org/ScratchResnickCreatividad.php].
---------- (2007b): ***All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten***; [Consulta en línea: Eduteka, Noviembre 9, 2009, http://web.media.mit.edu/~mres/papers/kindergarten-learning-approach.pdf].
Resnick, Mitchel y Otros (2009): ***Scratch: Progamming for all***; [Consulta en línea: Communications of the ACM, Noviembre 9, 2009, http://cacm.acm.org/magazines/2009/11/48421-scratch-programming-for-all/fulltext].
Ritter M., Grace Alexandra & Borja P., Raúl André (2000): ***Algoritmos y Programación***; Universidad Icesi, Cali.
Rizo Cabrera, Celia & Campistrous Pérez, Luis (2005): ***Didáctica y solución de problemas***; Ponencia presentada en el XIX Simposio Costarricense sobre Matemáticas, Ciencias y Sociedad [Consulta en línea: Universidad Nacional Heredia, Marzo 16 de 2007, http://www.cimm.ucr.ac.cr/simposios/recursos/XIX/campistrous.pdf]
Robinson, Ken (1999): ***All Our Futures: Creativity, Culture and Education***. National Advisory Committee on Creative and Cultural Education and DfEE publications, Inglaterra.
Rodríguez, Benjamín & Castro Walter (1995): ***Serie matemática construyamos 5***; Educar Editores, Bogotá.
Rojas A., Vicente & Ñacato C., José (1980): ***Técnica de flujogramas I***; Editora Andina, Quito, séptima edición.

Rumbaugh, James y Otros (1996): ***Modelado y diseño orientado a objetos, metodología OMT***; Prentice Hall, España.
Sabino, Carlos (1980): ***El proceso de investigación***, Ed. El Cid Editor, Bogotá.
Sánchez, Margarita A. de (1991): ***Desarrollo de habilidades de pensamiento***, Editorial Trillas, México, Decimocuarta reimpresión: 2004.
---------- (1993): ***Aprende a pensar, guía del instructor***; Editorial Trillas, México.
Savater, Fernando (1991): ***Ética para Amador***, Editorial Ariel, Barcelona.
---------- (1996): ***El valor de educar***; Editorial Ariel, Barcelona.
Serrano Pérez, Hugo (1998): ***Razonamiento abstracto***; Fénix impresores, Cali.
Sierra Vásquez, Francisco Javier (2001): ***La tecnología informática y la escuela***; [Consulta en línea: Ciberhabitat, Enero 8 de 2004, http://ciberhabitat.com/escuela/maestros/tiyescuela/ti_3.htm]
Schunk, Dale H. (1997): ***Teorías de aprendizaje***; Prentice Hall, México, segunda edición.
Solano, Guillermo (1991): ***Diseño lógico de exámenes***; Editorial Trillas, México.
Soto Sarmiento, Ángel Alonso (1997): ***Educación en tecnología, un reto y una exigencia social***; Cooperativa Editorial Magisterio, Bogotá.
Spoor, Cunera & Jinich, Emanuel (1990): ***Primeros pasos en Logo***; McGraw Hill, México.
Stager, Gary (2003): ***En pro de los computadores***; [Consulta en línea: Eduteka, Enero 13 de 2004, http://www.eduteka.org/ProComputadores.php]
Stenberg, Robert (1997): ***Inteligencia exitosa***. Paidos, España.
Treff, August & Jacobs, Donald (1983): ***Life skills mathematics***; Media Materials, Baltimore.
Trejos Buriticá, Omar Iván (1999): ***La esencia de la lógica de programación***; Editorial Papiro, Pereira.
Tucker, Allen –editor- (2003): ***A model currículum for K-12 computer science***, Final report of the ACM K-12 education task force curriculum committee; [Consulta en línea: ACM, Septiembre 12 de 2004, http://www.acm.org/education/K12]
Vasta, Ross y Otros (1996): ***Psicología infantil***; Ariel, Barcelona.
Watt, Daniel (1987): ***Aprendiendo con IBM Logo***; McGraw-Hill, Bogotá.
Wilson, James; Fernández, María & Hadaway, Nelda (1993): ***Technology and problem solving***; [Consulta en línea: Universidad de Georgia, Abril 6 de 2004, http://jwilson.coe.uga.edu/emt725/Pssyn/Pssyn.html]
Woolfolk, Anita E. (1999): ***Psicología educativa***; Prentice Hall, México, séptima edición.
Yarce, Jorge (2004): ***Valor para vivir los valores***; Editorial Norma, Bogotá.
Zea Restrepo, Claudia María y Otros (2000): ***Informática y escuela, un enfoque global (Conexiones)***; Editorial Universidad Pontificia Bolivariana, Medellín.
Zemelman, Steven; Daniels, Harvey; & Hyde, Arthur (1998): ***Mejores prácticas en matemáticas***; Editorial Heineman; [Consulta en línea: Eduteka, Julio 21 de 2004, http://www.eduteka.org/MejoresPracticas.php]
Zuleta Estanislao (1985): ***Educación y democracia***; Hombre Nuevo Editores, Medellín, quinta edición.
---------- (1996): ***Lógica y crítica***; Editorial Univalle, Cali.